



TADS: **Engenharia de Software I**

Carlos Eduardo Marquioni, PhD, PMP

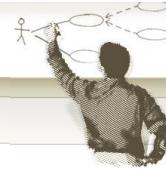
www.marquioni.com.br

Objetivos



- Esta disciplina visa qualificar/reciclar conhecimento relativo a:
 - Conceitos básicos da Engenharia de Software;
 - Ciclos de vida de software;
 - Engenharia de requisitos;
 - Qualidade de Software (utilizando o CMMI-DEV como referência).

Conteúdo Programático



- Este treinamento possui 4 módulos básicos:
 - **Módulo 1:** Engenharia de Software: Conceituação Básica;
 - **Módulo 2:** Ciclos de vida de software;
 - **Módulo 3:** Engenharia de requisitos;
 - **Módulo 4:** Qualidade de Software;
- 1º Bimestre: **Módulos 1, 2 e 3;**
- 2º Bimestre: **Módulo 4.**



Estudos dirigidos



- 1º Bimestre:
 - Pesquisa e desenvolvimento de trabalho escrito relacionado ao Processo XP.
- 2º Bimestre:
 - Pesquisa e desenvolvimento de 5 trabalhos escritos (contendo resumo e análise crítica) em relação às PA's VAL, VER, PPQA, PP e RD do CMMI-DEV v 1.3.





Módulo 1

Engenharia de Software: Conceituação Básica

www.marquioni.com.br

Realidade do Ambiente de Negócio

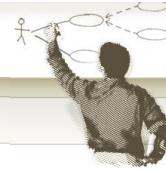


- Mudanças tecnológicas em aceleração (mais rápido, cada vez mais rápido);
- Globalização economia e privatização de empresas – Aumento da Concorrência;
- Desregulamentação (estrutura);
- Integração (conhecimento);
- Economia e Mercado Dinâmicos;
- Sistemas de Informação como principal recurso estratégico.



www.marquioni.com.br

Sobrevivência Empresarial



- Empresas de Sucesso
 - Reengenharia dos Processos de Negócio;
 - Qualidade Total;
 - Trabalhadores de Conhecimento;
 - Flexibilidade;
 - Inteligência Competitiva;
 - Empresas que Aprendem (*Learning Organizations*);
- Sistemas de informação podem apoiar na execução dos itens listados como condição para o sucesso;
- O desenvolvimento de Sistemas de informação tipicamente envolve desenvolvimento de produtos de software.

O que é Software?



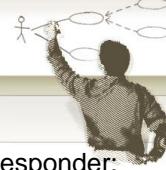
- É o resultado da Engenharia de Software, projetado e construído por profissionais de software (analistas e desenvolvedores);
- Software engloba:
 - Programas que são executados numa arquitetura previamente projetada;
 - Documentos (impressos ou em mídia eletrônica);
 - Dados (combinação de textos e figuras).

Engenharia de Software



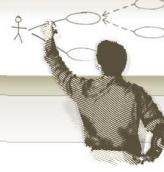
- IEEE: “É a aplicação sistemática, disciplinada e com abordagem quantitativa para o desenvolvimento; operação e manutenção de Software”;
- A Engenharia de Software engloba:
 - Processos - gerenciamento de requisitos, planejamento de projeto etc;
 - Um conjunto de Métodos “*how to's*” - mapeamento de processo de negócio, modelagem de dados, análise de sistemas, *design patterns* etc;
 - Conjunto de ferramentas – ferramentas para suportar os itens anteriores (p. ex., ferramentas CASE).

Visão Geral da Engenharia de Software



- A engenharia engloba entidades técnicas e sociais e deve responder:
 - Qual o problema a ser resolvido?
 - Quais as características técnicas e sociais que serão usadas para resolver o problema?
 - Como a entidade deve perceber a solução?
 - Como a solução será construída?
 - Qual será a abordagem usada para descobrir erros e falhas que aconteceram durante o projeto e a construção da solução?
 - Como a solução se comportará ao longo do tempo? Particularmente quando acontecerem as correções, adaptações e melhorias requisitadas pelos usuários? [elo com CM]

Complexidade e desenvolvimento de software



- O desenvolvimento de software é uma atividade complexa;
- Corresponde à sobreposição de complexidades relativas ao desenvolvimento de diversos componentes: software, hardware, procedimentos etc;
- Uma forma de abordar a complexidade e minimizar problemas envolve a definição de processos de software;
- Assim, um processo de desenvolvimento de software (ou processo de software ou metodologia de desenvolvimento) compreende as atividades necessárias para definir, desenvolver, testar e manter um produto de software;
- Há vários processos de software disponíveis no mercado (RUP, ICONIX, OPEN [Object-oriented Process, Environment and Notation], Volere, Freedom etc.

Paradigma

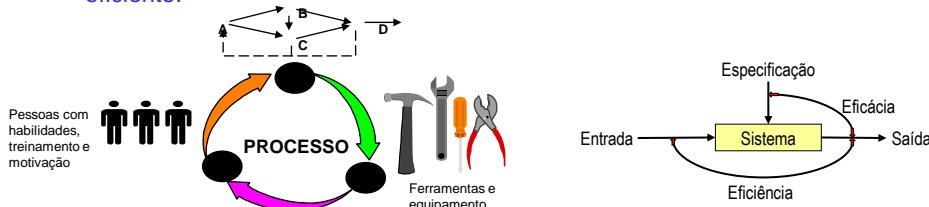


- Um *Paradigma* é uma forma de abordar um problema;
- Resumidamente, consideramos 3 paradigmas básicos para o desenvolvimento de software:
 - Análise e Projeto Estruturado (décadas 1970 – 1990);
 - Análise Estruturada;
 - Análise Estruturada Moderna (ou Essencial);
 - Análise e Projeto Orientado a Objetos (meados 1980 – atual);
 - Análise e Projeto Orientado a Aspectos (final 1990 – atual).

Definição de Processo de Software



- Processo
 - É uma sequência de passos executados para um determinado propósito (IEEE);
 - Processo de Software
 - É um conjunto de atividades, métodos, práticas e transformações que profissionais de informática utilizam para desenvolver e manter software e os produtos associados (CMM);
- Habilita a capacidade de repetir resultados, torna o desenvolvimento **eficaz** e **eficiente**.

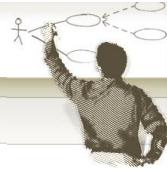


Processo de Software - Conteúdo



- O processo de software deve:
 - Possibilitar e orientar a utilização de **técnicas** de análise e projeto de sistemas;
 - **Definir:**
 - O que deve ser feito;
 - Quando;
 - Como;
 - Por quê;
 - Por quem;
 - Como será verificada a qualidade.

Porque se preocupar em definir processos?



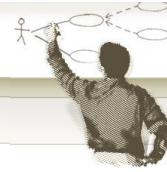
- No universo usuário...
 - “Ah! Lembrei de um *detalhezinho*: às vezes eu preciso imprimir isso. O relatório tem um gráfico, mas não lembro da fórmula. Quando preciso imprimir, é necessário informar outro campinho sobre o qual nós ainda não havíamos conversado, mas deve ser fácil de resolver com um *if*”;
 - “Ah! Mas isso é óbvio!”;
 - “Quanto falta para acabar?”
 - “Vai atrasar de novo?!!!”
- No universo dos profissionais de software...
 - “Já terminei 80%: só falta uma *coisinha*”;
 - “Agora sim: ficou intuitivo!”;
 - “Acho que preciso de um *tempinho a mais*”;
 - “Mas eu testei isso e *tinha* funcionado”;
 - “Como *parou de funcionar*? Funcionava até há pouco. E *ninguém mexeu nisso!*”;
 - “Terminei a manutenção! Mas *será que faltou* alterar alguma coisa?”

Características de um Processo Imaturo



- *Ad hoc* (casual) – improvisado pelos profissionais e pela gerência;
 - Técnicos se consideram artistas (eventualmente, Guerreiros Jedi);
- Não é rigorosamente seguido ou aplicado;
 - Padrões não existem, ou tendem a ser ignorados;
- Dependente dos profissionais atuais;
 - Ferramentas são usadas ao acaso, muitas vezes por iniciativa pessoal;
 - Metodologias praticadas informalmente;
- Baixa visibilidade de progresso e qualidade;
- A funcionalidade e a qualidade do produto podem ser comprometidas para atender ao cronograma;
- Uso arriscado de nova tecnologia;
- Custos excessivos de manutenção;
- Resultado (qualidade inclusive) difícil de prever;
- Entender relação: Maturidade pessoal x Maturidade Organizacional.

Características de um Processo Maduro



- Consistente com a forma como o trabalho é realizado;
- Definido, documentado e continuamente aprimorado;
- Entendido, utilizado e vivo na organização;
- Apoiado pelos níveis gerenciais;
- Utiliza métricas de produto e de processo;
- Novas tecnologias são introduzidas de forma disciplinada.

- Empresas maduras podem utilizar inclusive a imaturidade da concorrência (e do próprio cliente) como diferencial competitivo.

Organizações maduras melhoram a qualidade de vida dos profissionais



- Embora se saiba que os problemas tipicamente são originados pelos processos, e não pelas pessoas, uma organização imatura não tem esta visibilidade;
- Em uma organização madura:
 - As pessoas desenvolvem seu potencial de forma mais completa e são mais eficientes dentro da organização (papéis e responsabilidades definidas);
 - Através da definição, medição, e controle do processo, as melhorias são bem sucedidas e institucionalizadas;
 - Há probabilidade crescente de tecnologia, técnicas e ferramentas apropriadas estarem sendo introduzidas com sucesso.

Processos institucionalizados



- “Esta é a forma como fazemos as coisas por aqui”;
- A organização constrói uma infra-estrutura que contém processos efetivos, de fácil utilização e aplicados de forma consistente;
- A gerência apóia o uso: se ninguém se importa, todos esquecem;
- Processos institucionalizados permanecem depois que as pessoas que originalmente os definiram tiverem saído.

Processo de Software - Características



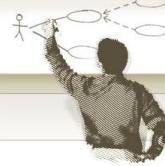
- Customizável:
 - Adaptável a qualquer **técnica**;
 - Adaptável a qualquer **tipo de projeto**;
- Versátil:
 - Aplicável a diversos **ciclos de vida**;
- Coerente:
 - Aderente às **ferramentas**, mas não dependente de ferramentas.

Para que Documentação/Formalização?



- Objetivo:
 - Auxiliar o processo de manutenção do software;
- Manutenção dos documentos:
 - Os produtos de trabalho gerados devem ser **mantidos**;
 - Qualquer alteração no software deve ser **refletida** na documentação;
- Ferramentas CASE:
 - **Facilitam** o armazenamento e manutenção.

Processo de Software - Fases

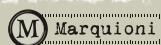


- Por que dividir o processo de software em fases?
 - O desenvolvimento de software é um **problema**;
 - Processo altamente complexo;
 - As fases são a decomposição do problema;
 - Por que decompomos?
 - É mais fácil **entender** vários problemas pequenos do que um único problema grande;
 - Tendo maior **visibilidade**, fica mais fácil encontrar a melhor solução e acompanhar a execução (gerenciar);
- ➔ DIVIDIR PARA CONQUISTAR.

Por que usar um Processo de Software?



- Facilidade de gerenciamento:
 - **Visibilidade:**
 - Saber o que, quando e por quem deve ser feito;
 - Saber quanto foi feito, quanto falta;
- Documentação padronizada:
 - Menor **dependência** pessoal:
 - Manutenção pode ser realizada por outro, que não o “pai” do sistema;
 - Facilidade de **comunicação**;
- Aumento de qualidade:
 - Processo padronizado e controlado;
- Aumento de produtividade:
 - Documentação padronizada;
 - Não é necessário “reinventar a roda”.



www.marquioni.com.br



Módulo 2

Ciclos de vida de software

www.marquioni.com.br

Ciclos de Vida



- Partes que compõe a vida de um objeto em estudo:
 - Composto por fases;
- Exemplos:
 - Ciclo de vida humano:
 - Nascimento, Crescimento, Morte;
 - Ciclo de vida do software ou sistema:
 - Concepção, Desenvolvimento, Maturação e Desativação;
 - Ciclo de vida de desenvolvimento de software:
 - EV, ER, DA, DD, Co, Te [TU, TI, TS, TA], Im, Pl;
 - EV, RF, RS, DA, DD, Co, Testes, Im, Pl;
- Processos de Software definem inclusive o ciclo de vida de desenvolvimento de software, e especificam como executar estas fases.

Definições para Projetos de Software



- Projeto de Desenvolvimento:
 - Normalmente parte de um conceito;
 - Complexidade: pequena, média ou alta;
 - Tamanho: pequeno, médio ou grande;
- Projeto de Manutenção:
 - Normalmente parte de um produto (software) já existente;
 - Complexidade: pequena, média ou alta;
 - Tamanho: pequeno ou médio.

Fases de Ciclo de Vida de Engenharia de Software



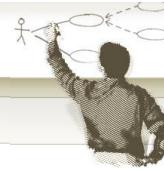
- Fases “padrão” para um ciclo de vida de Engenharia de Software:
 - Estudo de Viabilidade (EV);
 - Estudo de Requisitos (ER);
 - Desenho Arquitetônico (DA);
 - Desenho Detalhado (DD);
 - Codificação (Co);
 - Testes (Te)
 - Teste de Unidade (TU);
 - Teste de Integração (TI);
 - Teste de Sistema (TS);
 - Teste de Aceite (TA);
 - Implantação (Im);
 - Pós-Implantação (PI).
- Observe que as fases listadas são exemplos “Padrão” – cada processo de software pode renomear suas fases de acordo com o interesse da organização;
- Ocorre que tipicamente, embora os nomes variem, o conteúdo executado é relativamente parecido;
- O ciclo de vida de manutenção é derivado do ciclo de vida de desenvolvimento.

Estudo de Viabilidade (EV)



- **Objetivo:**
 - **Levantar** alternativas/Verificar viabilidade do produto ou da manutenção;
- **Descrição**
 - Tipicamente, pré-venda [tanto em projetos internos quanto contratados de terceiros];
 - **Levantamento inicial** dos requisitos de modo a viabilizar um **estudo de alternativas** e uma **primeira estimativa** para o desenvolvimento ou manutenção:
 - Exemplos de alternativas:
 - Desenvolvimento interno;
 - Terceirização;
 - Compra de pacote;
 - Fatores que influenciam na definição de alternativas:
 - Técnicos (tecnologia exigida, profissionais,...);
 - Não Técnicos (\$, prazo,...).

Estudo de Viabilidade (EV)



- Artefatos Típicos
 - Diagramas de Mapeamento de Processos de Negócio;
 - Relatório de Viabilidade (via de regra, Proposta técnica/comercial):
 - Com as alternativas encontradas e seus custos associados;
 - Contrato:
 - Com as alternativas propostas e seus custos associados.

Estudo de Requisitos (ER)



- Objetivo
 - Definir/formalizar o problema;
- Descrição
 - Mapeamento (junto aos usuários) do negócio e das funcionalidades a serem implementadas;
 - “O que” na ótica do usuário;
- Artefatos Típicos
 - Documento de Requisitos do Usuário (DRU);
 - Diagramas Técnicos de Software (*Use Cases* em formato estruturados p. ex.);
 - MER conceitual.

Desenho Arquitetônico (DA)



- Objetivos
 - Solucionar/formalizar a solução do problema;
- Descrição
 - Determinar como o software implementa as funcionalidades requeridas;
 - “Como” fazer;
- Artefatos Típicos
 - Documento de Desenho Arquitetônico (DDA) [às vezes referenciado como Documento de Arquitetura];
 - Diagramas Técnicos de Software (Classes de Análise [Domínio e por Caso de Uso], Diagrama de Sequência, p. ex.)
 - MER Lógico.

Desenho Detalhado (DD)



- Objetivos
 - Detalhar tecnicamente a solução encontrada;
- Descrição
 - Detalhamento do desenho arquitetônico para especificação dos programas e objetos a serem construídos;
- Artefatos Típicos
 - Especificação dos componentes;
 - Diagramas técnicos de software (Diagrama de Componentes, Distribuição etc);
 - MER Físico.

Codificação (Co)



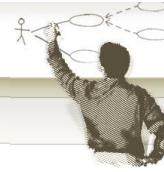
- Objetivos
 - Implementar a solução;
- Descrição
 - Criação dos programas e objetos com base nas especificações;
- Artefatos Típicos
 - Script de criação do banco de dados;
 - Banco de dados criado (estrutura física) no SGBD utilizado;
 - Programas e objetos criados/compilados.

Teste de Unidade (TU)



- Objetivo
 - Validar cada programa isoladamente;
- Descrição
 - Validação [enquanto adequação ao uso] do funcionamento básico adequado dos programas em relação às especificações;
 - Teste “caixa branca”;
- Artefatos Típicos
 - Resultado dos Testes de Unidade;
 - Relatório de problemas encontrados.

Teste de Integração (TI)



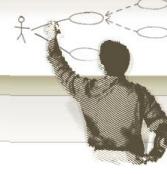
- Objetivo
 - Validar as interfaces entre programas;
- Descrição
 - Validação [enquanto adequação ao uso] da correta interação dos programas, conforme as especificações;
- Artefatos Típicos
 - Resultado dos Testes de Integração;
 - Relatório de problemas encontrados.

Teste de Sistema (TS)



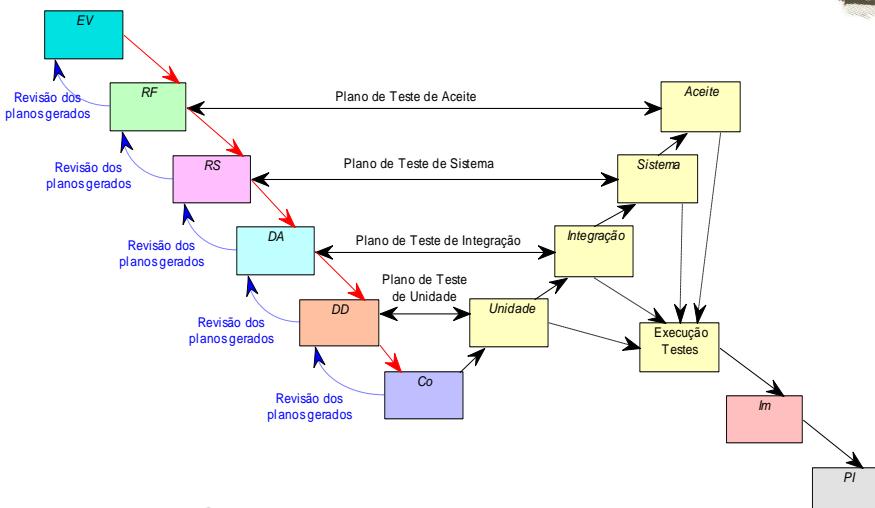
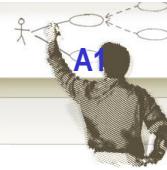
- Objetivo
 - Validar a execução do produto no ambiente;
 - Tipicamente envolve a instalação do produto em ambiente monitorado (QA);
- Descrição
 - Validação [enquanto adequação ao uso] da correta interação do software considerando:
 - O sistema operacional;
 - O hardware;
 - O firmware;
 - Outros aplicativos.
- Artefatos Típicos
 - Resultado dos Testes de Sistema;
 - Relatório de problemas encontrados.

Teste de Aceite (TA)

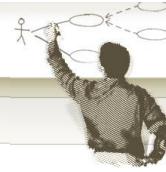


- Objetivo
 - Validar as funcionalidades junto aos usuários (obter o aceite do produto);
- Descrição
 - Validação [enquanto adequação ao uso] do atendimento do software quanto às funcionalidades solicitadas;
 - Teste “caixa preta”;
- Artefatos
 - Resultado dos Testes de Aceite;
 - Relatório de problemas encontrados;
 - Termo de Aceite assinado;
 - Manual de Operação;
 - Manual de Instalação.

Testes em “V”



Implantação (Im)



- Objetivos
 - Instalar o software;
- Descrição
 - Disponibilizar o software em produção, e acompanhar os primeiros ciclos de uso;
- Artefatos
 - Manual do Usuário.
 - Documento de Transferência de Software assinado;
 - Programas instalados no ambiente de produção.

Pós-Implantação (PI)



- Objetivos
 - Avaliar satisfação dos usuários;
- Descrição
 - Avaliação do nível de satisfação dos clientes/usuários com o produto entregue;
 - Análise de retorno de investimento (ROI);
 - Revisão dos requisitos não implementados;
 - Levantamento de novas necessidades (elo/ciclo com EV);
- Artefatos
 - Resultado da Pesquisa de Satisfação;
 - Resultado da Análise de Investimento;
 - Sugestão de Requisitos para a próxima release do produto.

Complexidade de Projeto: Critério de seleção do ciclo de vida



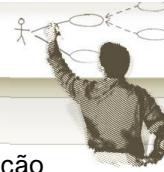
- Para pensar complexo, integrado, de forma sistêmica as condições de aplicabilidade dos ciclos de vida, é necessário utilizar critérios objetivos para definir o ciclo de vida a utilizar em cada projeto;
- A engenharia pressupõe análise e reflexão antes de aplicar o mesmo ciclo de vida padrão em qualquer situação.
- Assim, define-se (apenas como exemplo) que a complexidade de um projeto é definida em função de três variáveis:
 - **Tipo de problema:**
 - Não estruturado (requisitos indefinidos ou instáveis);
 - Estruturado (requisitos definidos ou estáveis);
 - **Tamanho/esforço do projeto** [atenção para a diferença entre tamanho e esforço!!]:
 - Pequeno (até 100 unidades de tamanho) → Esforço < 160 HH;
 - Médio (entre 101 e 1000 unidades de tamanho) → Esforço entre 160 e 600HH;
 - Grande (mais que 1001 unidades de tamanho) → Esforço >1000 HH;
 - **Tecnologia envolvida:**
 - Conhecida;
 - Nova.

Exemplos de Ciclos de Vida



- A seguir são debatidos os seguintes ciclos de vida, considerando as variáveis de complexidade do slide anterior:
 - Cascata Simples;
 - Cascata com Sobreposição;
 - Crítico no Tempo;
 - Versionamento – Incremental;
 - Versionamento – Time Box;
 - Prototipação Rápida;
 - Paralelismo;
 - USDP – RUP.

Fases do Ciclo de Vida



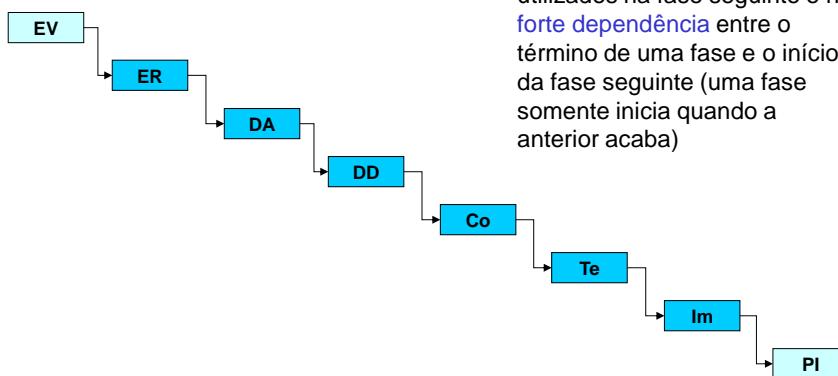
- Por simplicidade, vamos utilizar as fases comentadas na seção anterior:
 - Estudo de Viabilidade (EV);
 - Especificação/Engenharia de Requisitos (ER);
 - Desenho Arquitetônico (DA);
 - Desenho Detalhado (DD);
 - Codificação (Co);
 - Testes (Te);
 - Implantação (Im);
 - Pós-Implantação (Pl).

Cascata Simples

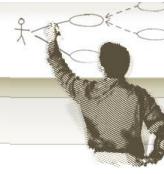


- Características

- Os produtos de uma fase são utilizados na fase seguinte e há forte dependência entre o término de uma fase e o início da fase seguinte (uma fase somente inicia quando a anterior acaba)



Cascata Simples



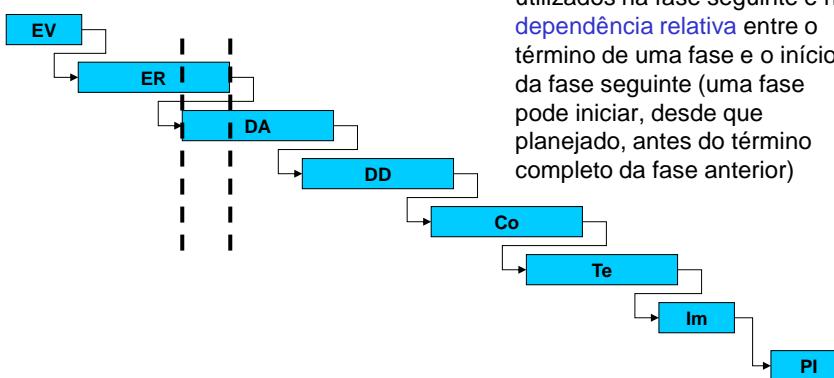
- Aplicabilidade
 - Tipo de Problema: **estruturado**;
 - Requisitos bem definidos;
 - As necessidades do usuário são conhecidas;
 - Tipicamente aplicável para automatização de processos já definidos e documentados, e sem alteração de processo prevista;
 - Tamanho do Projeto: **pequeno ou médio**;
 - Interessante principalmente no caso de pequenas manutenções;
 - Tecnologia utilizada: **conhecida**;
- Recomendações / Observações
 - Fácil visibilidade e controle;
- Problema
 - **Alta dependência** de uma fase para outra;
 - Atrasos também tendem a ocorrer “em cascata”.

Cascata com Sobreposição



• Características

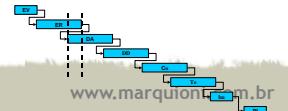
- Os produtos de uma fase são utilizados na fase seguinte e há **dependência relativa** entre o término de uma fase e o início da fase seguinte (uma fase pode iniciar, desde que planejado, antes do término completo da fase anterior)



Cascata com Sobreposição



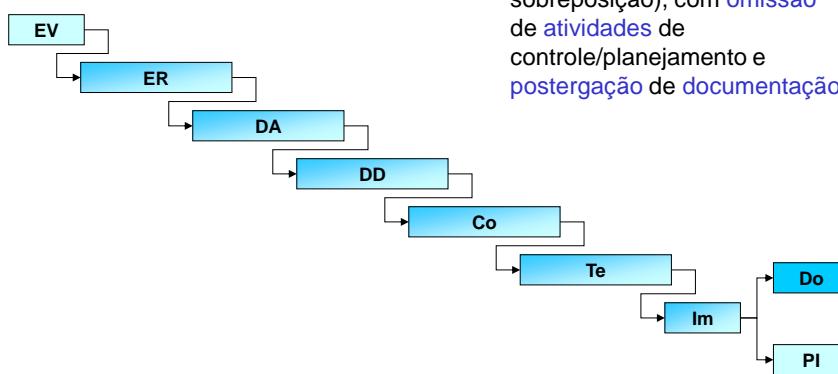
- Aplicabilidade
 - Tipo de Problema: **estruturado**;
 - Tamanho do Projeto: **pequeno, médio ou grande**;
 - Tecnologia utilizada: **conhecida**;
- Recomendações / Observações
 - Pode reduzir o **tempo de entrega** (caso não haja retrabalho);
 - Flexibiliza **alocação** de recursos;
- Problema
 - Grande probabilidade de **retrabalho**;
 - Se o retrabalho for MUITO grande, torna a estratégia ineficiente;
 - Risco de sobrepor mais de uma fase
 - Aumenta a probabilidade de retrabalho;
 - Risco de **documentação incompleta** ou desatualizada
 - Excesso de alterações;
 - Dificulta a visibilidade e controle.



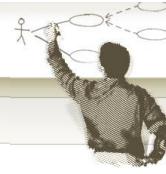
Crítico no Tempo



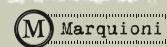
- Características
 - Cascata (simples ou com sobreposição), com **omissão** de **atividades** de controle/planejamento e **postergação** de **documentação**



Crítico no Tempo



- Aplicabilidade
 - Tipo de Problema: **estruturado**;
 - Tem que se saber o que se quer;
 - Tamanho do Projeto: **pequeno, médio ou grande** – uma vez que se trata de projeto com dependência crítica de tempo (ex.: requisito legal);
 - Tipo de Tecnologia: **conhecida**;
- Recomendações / Observações
 - “Só quebrar em caso de emergência”;
 - Somente deve ser utilizado como exceção (nunca como regra);
 - Deve ter a autorização prévia de uma comissão executiva;
 - PLANEJAR e REALIZAR a atualização do conteúdo de especificação (criar uma fase “Do - Documentação” e garantir que seja executada ao final).

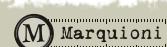


www.marquioni.com.br

Crítico no Tempo

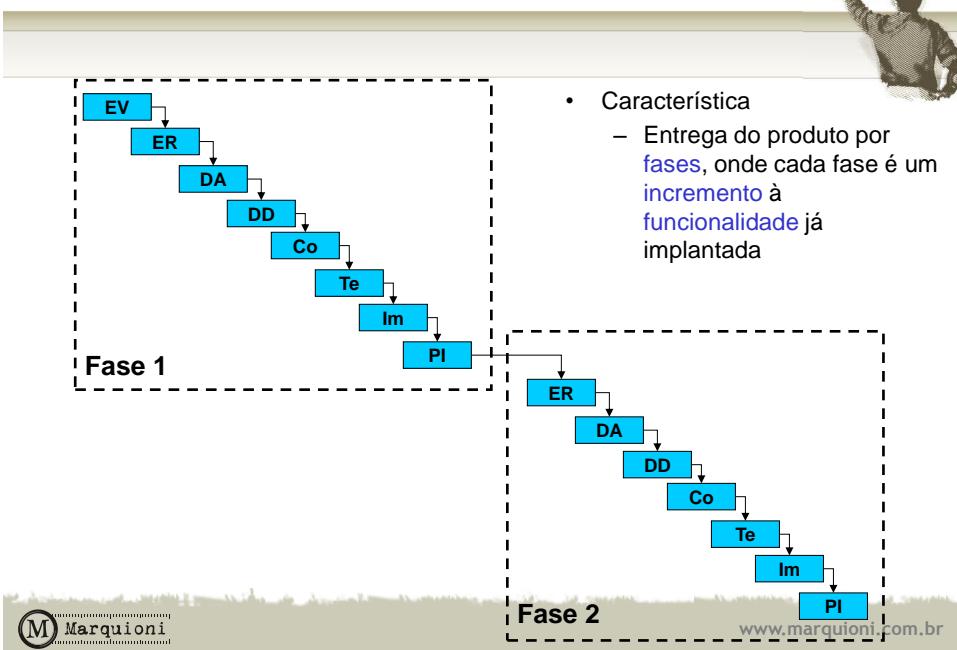


- Problemas
 - Qualidade prejudicada
 - A pressa e a falta de controle aumentam a probabilidade de introdução de erros;
 - Perda de controle
 - A utilização sem o devido critério pode institucionalizar o caos;
 - Perda de documentação
 - Há grande probabilidade da atualização da especificação nunca ser executada ou ter sua qualidade prejudicada (fase “Do” ser “deixada para depois”);
 - Aumento de custo
 - Documentação *a posteriori* mais cara.

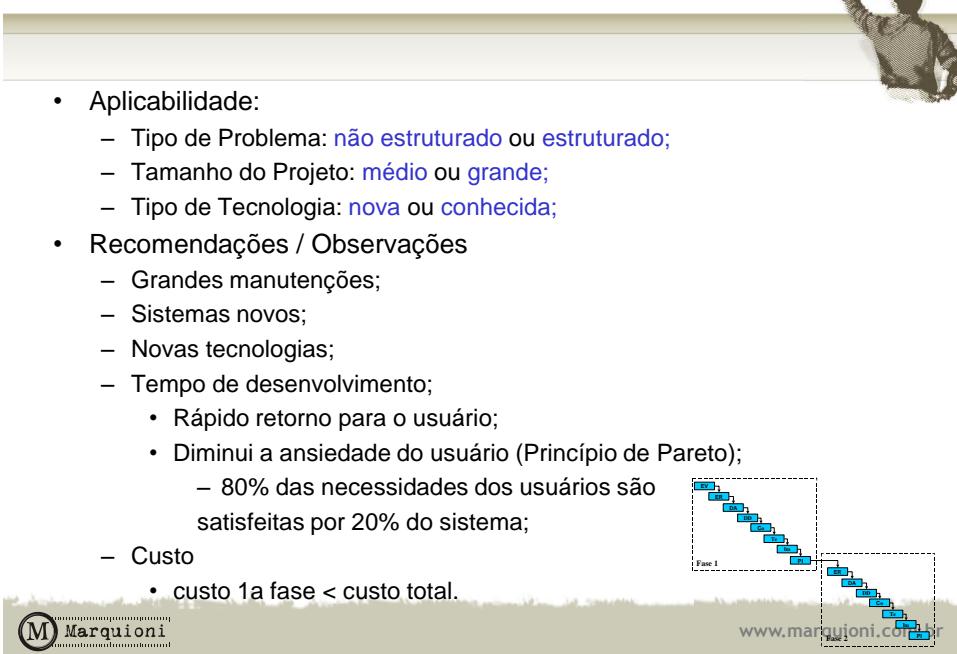


www.marquioni.com.br

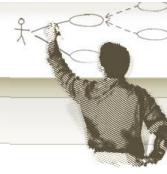
Versionamento - Incremental



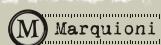
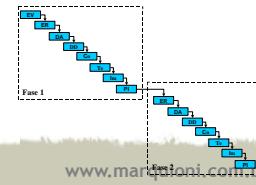
Versionamento - Incremental



Versionamento - Incremental



- Recomendações / Observações (cont.)
 - Fácil de controlar (*phase-driven*);
 - Visão
 - É preciso ter a visão completa do produto;
 - A **arquitetura global** deve estar pronta para a construção da 1a. Fase;
 - A 1a. fase deve implementar a infra-estrutura para as seguintes.

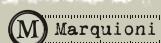
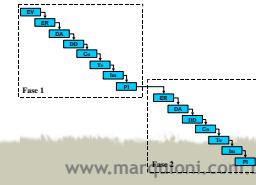


www.marquioni.com.br

Versionamento - Incremental

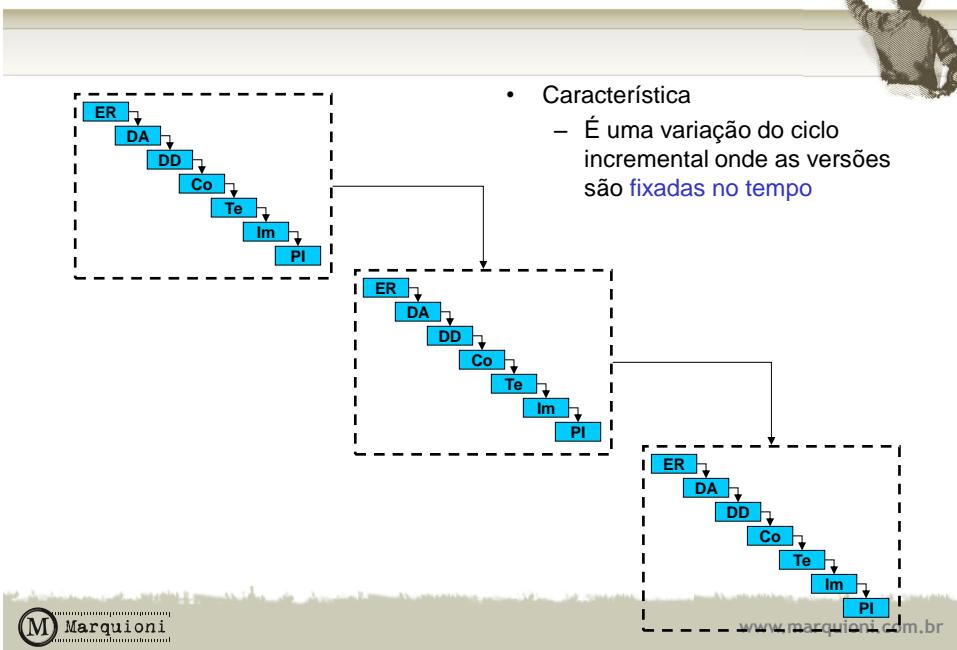


- Problemas
 - Visão
 - Dificuldade de definir a arquitetura global;
 - A infra-estrutura pode “pesar” muito na 1a fase;
 - Retrabalho
 - A cada nova fase a fase anterior deve ser revisada e, possivelmente, alterada;
 - Soluções intermediárias jogadas fora;
 - Tempo de desenvolvimento
 - Tempo total > tempo não *faseado*;
 - Custo
 - Custo total > custo não *faseado*.

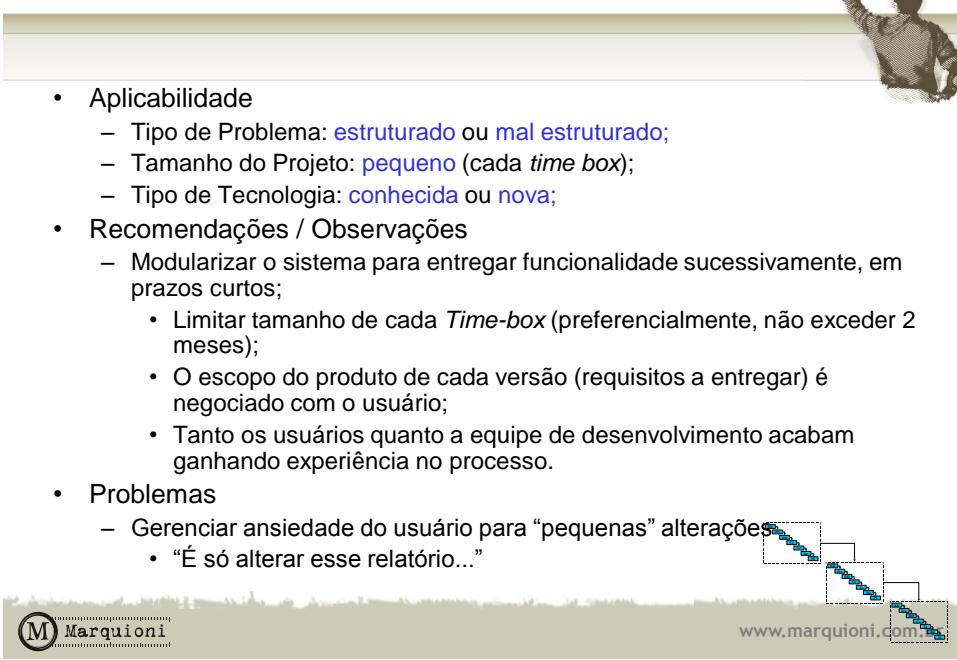


www.marquioni.com.br

Versionamento - Time Box



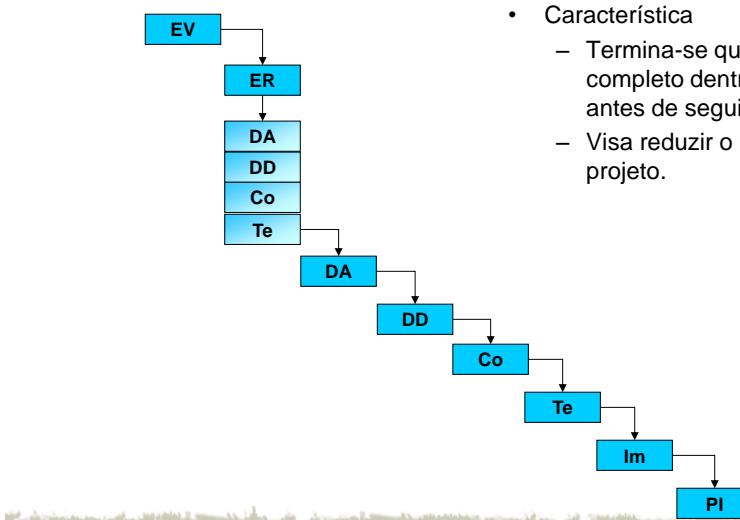
Versionamento - Time Box



Prototipação Rápida



- Característica
 - Termina-se quase um ciclo completo dentro de uma fase, antes de seguir para a próxima;
 - Visa reduzir o custo ou risco do projeto.



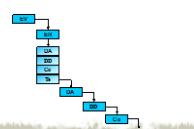
M Marquioni

www.marquioni.com.br

Prototipação Rápida



- Aplicabilidade
 - Tipo de Problema: **estruturado** ou **não estruturado**;
 - Tamanho do Projeto: **pequeno**, **médio** ou **grande**;
 - Tipo de Tecnologia: **conhecida** ou **nova**;
 - Recomendações / Observações
 - É inclusive uma técnica para **levantamento** de requisitos e estudo de alternativas;
 - Melhora o entendimento dos requisitos e descobre novos requisitos;
 - Utilização
 - Pode ser utilizado em qualquer fase do desenvolvimento;
 - Problemas
 - Negligenciar especificação do produto
 - O protótipo se tornar a “especificação do produto”.



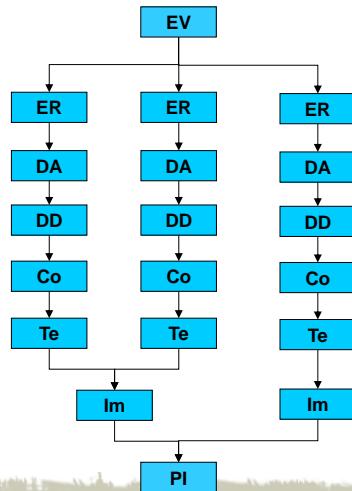
M Marquioni

www.marquini.com.br

Paralelismo



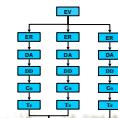
- Característica
 - O projeto é **dividido** entre diversas equipes e desenvolvido **concorrentemente**



Paralelismo



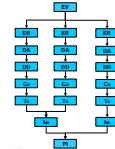
- Aplicabilidade
 - Tipo de Problema: **estruturado**;
 - Tamanho do Projeto: **médio ou grande**;
 - Tipo de Tecnologia: **conhecida**;
- Recomendações / Observações
 - Somente quando é **possível separar** claramente os módulos dos sistemas em grupos independentes;
 - Variações
 - Os ciclos anteriormente descritos se aplicam a cada uma das ramificações deste SDLC [Software Development Life Cycle];
 - Subcontratação;
 - Reduz o **tempo de entrega** do projeto.



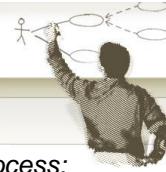
Paralelismo



- Problemas
 - Gerenciamento das atividades paralelas e sincronização das convergências (sincronização dos trabalhos);
 - Aumento dos riscos do projeto;
 - Nem tudo pode ser realizado em paralelo:
 - *Nove tecelões fazem um tapete em 1 mês; mas*
 - *Nove grávidas NÃO fazem um filho em 1 mês.*
 - *E no caso do tapete, o trabalho dos tecelões necessita ser “integrado” antes de o produto ser considerado concluído.*



USDP/RUP (1 de 3)



- *Unified Software Development Process/Rational Unified Process;*
- Também chamado de Ciclo de Vida de Processo Incremental e Iterativo;
- Processo de engenharia e gestão de software;
- Dividido em Disciplinas:
 - Modelagem de Negócio;
 - Requisitos;
 - Análise e design;
 - Implementação;
 - Testes;
 - Implantação;
 - Gerência de configuração e mudança;
 - Gerenciamento de Projeto;
 - Ambiente.

USDP/RUP (2 de 3)

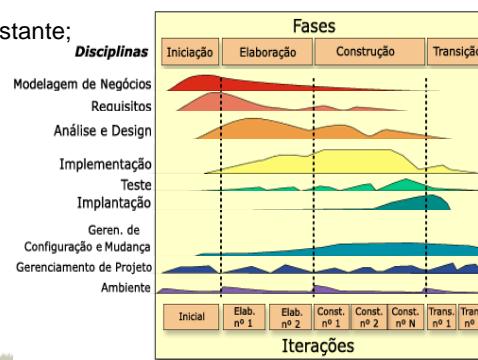


- Dividido em Disciplinas:
 - Modelagem de Negócio;
 - Requisitos;
 - Análise e design;
 - Implementação;
 - Testes;
 - Implantação;
 - Gerência de configuração e mudança;
 - Gerenciamento de Projeto;
 - Ambiente;
- Dividido em Fases (dimensão temporal):
 - Iniciação;
 - Elaboração;
 - Construção;
 - Transição.

USDP/RUP (3 de 3)



- Define como boas práticas:
 - Desenvolvimento iterativo;
 - Gestão de requisitos (RMUC);
 - Uso de arquitetura de componentes;
 - Modelagem visual (UML);
 - Verificação de qualidade constante;
 - Gestão de mudança.



Características



- Divide o desenvolvimento de um produto de software em ciclos (iterações):
 - Em cada ciclo são executadas as Disciplinas de análise, projeto, implementação e testes (o que varia é o esforço aplicado);
 - Cada um dos ciclos considera um subconjunto de requisitos;
 - No próximo ciclo, outro subconjunto é considerado (o que produz um novo incremento do sistema em relação à versão anterior);
 - Cada ciclo é executado através de uma “minicascata”;
 - Aplicável se existir um mecanismo para dividir os requisitos do sistema em partes, para que cada parte seja alocada a um ciclo do desenvolvimento:
 - Critérios de seleção de requisitos: importância para o negócio e risco de cada requisito;
 - Maiores riscos são resolvidos nas primeiras iterações;
 - Problema: aumento na dificuldade de gestão.

Dimensões



- O ciclo de vida de processo incremental e iterativo pode ser estudado segundo 2 dimensões:
 - Dimensão Temporal (Iterações):
 - Estrutura o processo em fases; em cada uma dessas fases há uma ou mais iterações;
 - Cada iteração tem uma duração preestabelecida (2 a 6 semanas);
 - Dimensão de Atividades (Disciplinas):
 - Atividades realizadas durante a iteração (o que varia é o esforço aplicado em cada Disciplina para cada iteração);
 - Em cada Fase, diferentes artefatos são gerados, ou artefatos das fases anteriores são revisados/estendidos com mais detalhes.

Fases do Processo Unificado (1 de 2)



- Iniciação
 - Idéia geral e o escopo do desenvolvimento são desenvolvidos;
 - Planejamento de alto nível;
 - Determinação dos marcos que separam as fases;
- Elaboração
 - Alcançado entendimento inicial sobre como construir o sistema;
 - Completado planejamento inicial;
 - Análise de domínio do negócio;
 - Requisitos alocados às iterações (importância e risco);
 - Planejamento macro próximas iterações;

Fases do Processo Unificado (2 de 2)



- Construção
 - Aumento das atividades de análise e projeto;
 - Fase em que ocorrem mais iterações incrementais;
 - Se há entrega para o usuário final., é elaborado manual do usuário;
- Transição
 - Usuários são treinados no uso do sistema;
 - Tratadas questões de instalação e configuração;
 - Aceite do usuário e análise de gastos.

Referências básicas



- O ciclo de Vida USDP/RUP possibilita elos evidentes com outros ciclos de vida estudados:
 - Cascata;
 - Versionamento – Incremental;
 - Versionamento - Time Box;
 - Aplicável nas condições destes ciclos de vida com elo evidente;
- Elaborado a partir da referência Objectory.



www.marquioni.com.br



Módulo 3

Engenharia de requisitos

www.marquioni.com.br

Requisito - Conceito



- “Um requisito é algo que o produto deve obrigatoriamente fazer ou uma qualidade que ele deve obrigatoriamente possuir. Um requisito existe tanto porque o tipo de produto demanda certas funções ou qualidades quanto porque o cliente quer aquele requisito como parte do produto entregue” (ROBERTSON; ROBERTSON, 2007, p. 09).

Requisito Funcional - Conceito



- “Um requisito funcional é uma ação que o produto deve executar para ser útil aos usuários. Requisitos funcionais surgem a partir do trabalho que seus *stakeholders* necessitam fazer. Quase toda ação – calcular, inspecionar, publicar [...] – pode ser um requisito funcional. [...] Este requisito é algo que o produto obrigatoriamente deve fazer para ser útil no contexto do negócio do cliente” (ROBERTSON; ROBERTSON, 2007, p. 09).

Engenharia de Requisitos



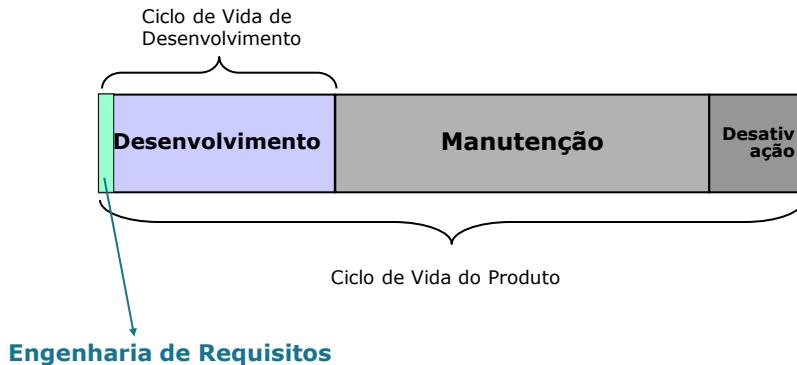
- A Engenharia de Requisitos corresponde ao uso de técnicas sistemáticas e repetíveis para garantir que os requisitos do software estejam completos e consistentes:
 - Existe um *quase consenso* em relação aos processos que devem ser executados para atuar com Engenharia de Requisitos.

Quando Aplicar?

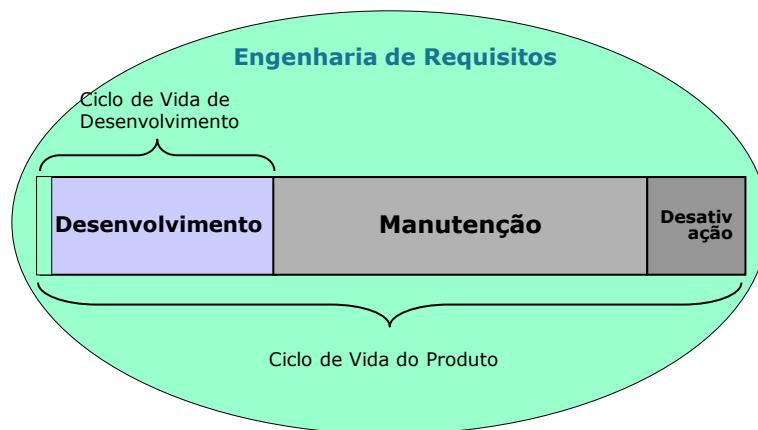


- A engenharia de requisitos não é uma atividade exclusiva do início do projeto de software;
- Tem aplicação iniciada no começo de um projeto, que **se estende durante todo o ciclo de vida do software**:
 - A engenharia de requisitos começa com o início do projeto de desenvolvimento e só termina quando o produto de software é desativado;
 - Os profissionais podem pensar se tratar de uma sequência linear de atividades: esta é uma visão simplificada do processo.

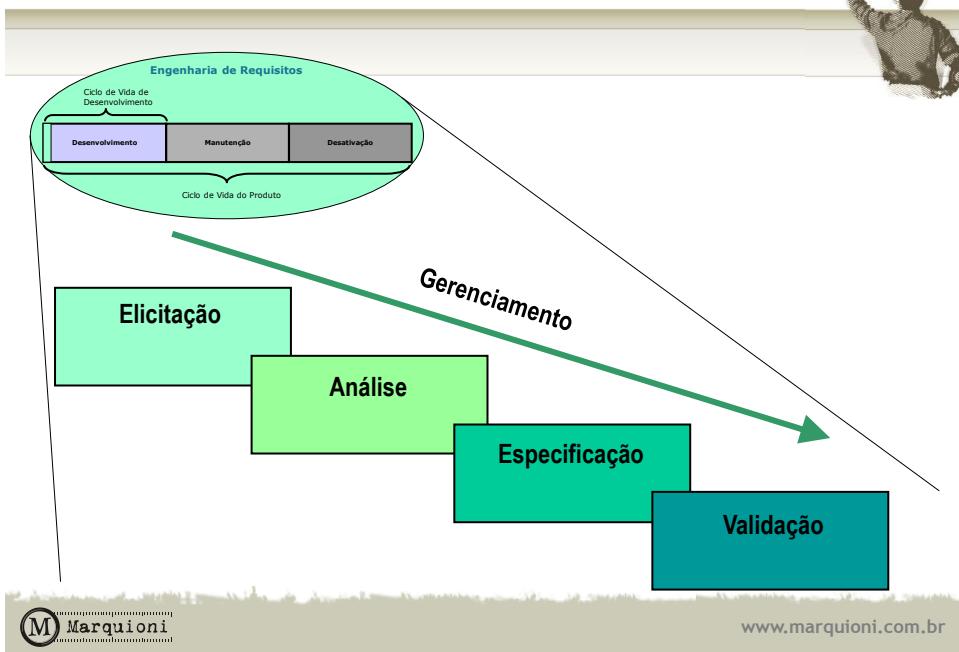
Engenharia de Requisitos x Ciclo de Vida do Software



Engenharia de Requisitos x Ciclo de Vida do Software



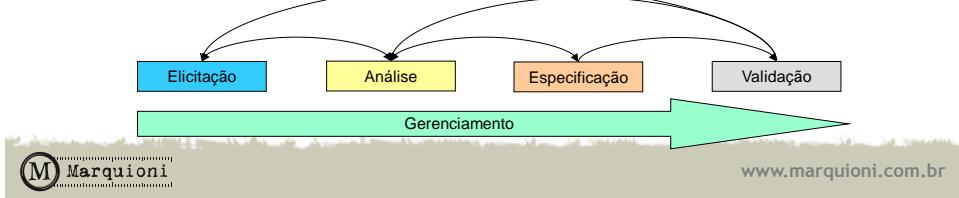
Os Processos da Engenharia de Requisitos



Marquioni

www.marquioni.com.br

Engenharia de Requisitos - Geral



Marquioni

www.marquioni.com.br

Os Processos da Engenharia de Requisitos



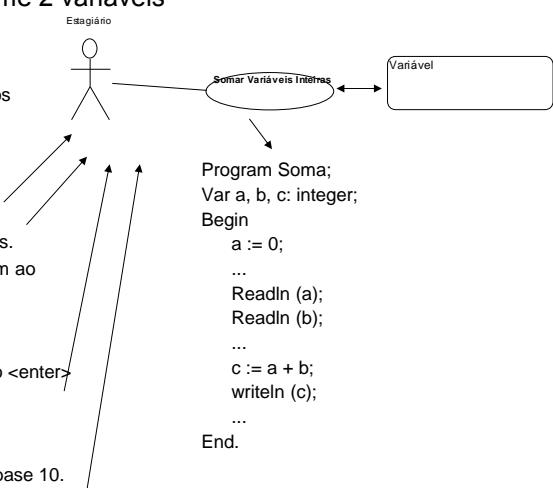
- Mas como esses 5 processos básicos influenciam o fato que a Engenharia de Requisitos deve ser aplicada durante o ciclo de vida do software, e não apenas no início do desenvolvimento?

→ Desenvolver software é transcrever requisitos...

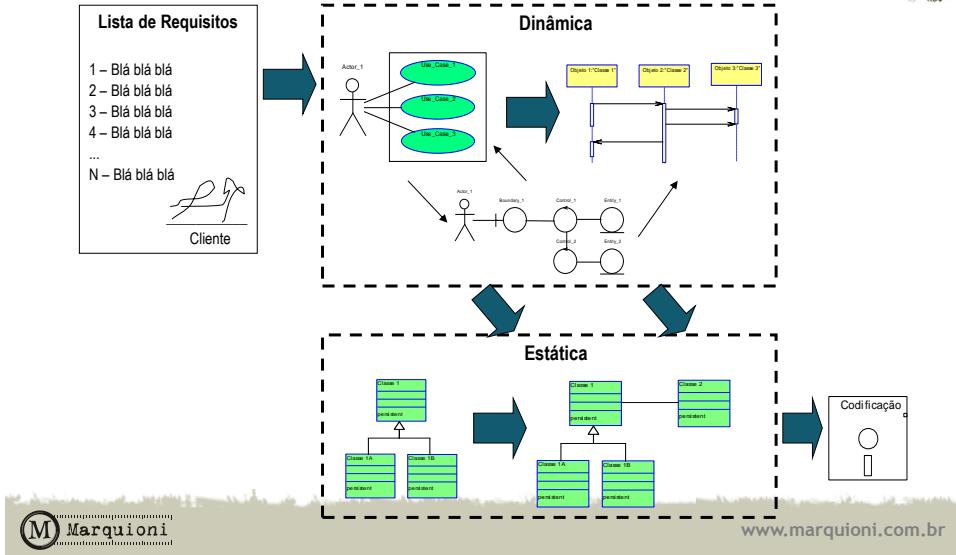
As traduções dos requisitos

“Eu quero um sistema que some 2 variáveis”

- **RF-int1** – Interface para soma.
O sistema tem tela para somar 2 números informados. São campos desta tela:
 - Valor variável 1;
 - Valor variável 2;
 - Valor soma ($1 + 2$).
- **RF-neg1** – Conjunto de números válidos.
O sistema soma variáveis que pertençam ao conjunto Z (números inteiros).
- **RF-neg2** – Momento da soma.
A soma só acontece após pressionado o <enter> relativo ao segundo número informado.
- **RF-neg3** – Base 10.
O sistema efetua soma de números na base 10.



Variantes de Requisitos ao Longo do Desenvolvimento



Considerações Finais



- Em projetos reais:
 - Dificilmente vou me deparar com situações simples o suficiente para que sejam resolvidas sem necessitar do apoio de aspectos teóricos e conceituais;
- Em organizações sérias:
 - Tentar abordar temas complexos utilizando apenas *feeling* ou intuição pode ‘fritar’ o profissional (tanto quando se tratar de um consultor quanto um profissional interno);
- Um profissional que se destaca é aquele que:
 - Possui conhecimento teórico e conceitual;
 - Consegue aplicar estes conhecimentos para resolver problemas reais.



Módulo 4

Qualidade de Software

www.marquioni.com.br

Qualidade de software – Algo *natural*

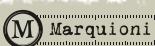


Philip Crosby comenta que:

“O problema da qualidade não está associado ao que as pessoas não sabem sobre ela, mas ao que elas acham que sabem. Neste sentido, qualidade tem semelhanças com sexo:

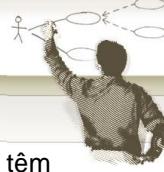
- Todos estão preparados para ele (em certas condições, é óbvio);
- Todos sentem que entendem do assunto (até pensam não ser necessário explicar o assunto);
- Todos pensam que sua execução está associada a seguir instintos naturais;
- E, óbvio, a maioria das pessoas crê que os problemas nesta área são causados pelos outros.”

(ap. Pressman, 2000, p. 192).



www.marquioni.com.br

Qualidade de software – Aspecto Onipresente



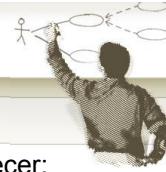
- Os papéis envolvidos no processo de software afirmam que têm preocupação com qualidade:
 - Desenvolvedores: “é claro que meus programas têm qualidade”;
 - Testadores: “vou avaliar a qualidade”;
 - Gerentes: “nossa preocupação maior é com a qualidade”;
- Contudo:
 - Raramente há planejamento da qualidade;
 - Raramente a qualidade é avaliada de forma objetiva (inclusive pelos testadores – “agora vou dar uma ‘testada’”).

Como ter qualidade no desenvolvimento de software?



- Não é suficiente ter preocupação com a qualidade ou dizer que qualidade é importante; é necessário:
 - Definir explicitamente o que significa qualidade de software;
 - Criar um conjunto de atividades para evidenciar que os artefatos gerados ou atualizados pela engenharia de software seguiram um nível de qualidade definido e acordado;
 - Usar métricas para desenvolver estratégias de melhoria no processo de software para, como consequência, aumentar a qualidade do produto final...

Valor da Qualidade



- A noção de “qualidade” não é tão simples quanto pode parecer:
 - Existem muitos **desejos** de qualidade relativos a uma perspectiva particular do produto (usuários diferentes entendem qualidade de forma diferente);
 - Características de qualidade podem ser requeridas ou não, ou podem ser requeridas em um nível maior ou menor; pode haver *negociação* do nível de qualidade desejado;
 - Os profissionais técnicos e gestores do projeto devem ser capazes de apresentar alternativas para negociação da qualidade...

Produção de software em organizações imaturas



- A produção de software em uma empresa com processos imaturos costuma ser resumida como:
 - Os requisitos são recebidos;
 - Um produto de software é (geralmente) produzido através de algum processo indeterminado;
 - Um produto é gerado e (com sorte) funciona.

Entra    Sai

Contatos com melhoria de qualidade convencional

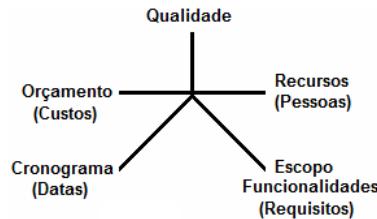


- A melhoria foca na correção do processo, não em procurar culpados;
- A melhoria é mensurada e periodicamente reforçada;
 - “In God we trust, all other bring data” – Deming;
- A melhoria é um processo contínuo;
- Se o nível de desconforto não está alto o suficiente, as coisas não costumam mudar.

Restrições a considerar



- A equipe do projeto deve estar atenta às restrições relacionadas ao projeto, e nas relações entre elas:
 - Qualidade;
 - Escopo;
 - Cronograma;
 - Orçamento;
 - Recursos;
- Dica: utilizar as restrições para análise de *potencial de negociação*.

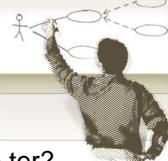


Exemplo: A relação qualidade x custo



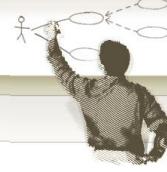
- Normalmente o cliente não está atento à relação qualidade x custo:
 - A qualidade é “premissa”: porque pagar a mais por ela?
 - Critérios objetivos auxiliam a evidenciar se tratar de aspecto que envolve negociação, como em outras engenharias (p. ex., a qualidade do acabamento na construção civil provoca variação de valores).

Negociação da qualidade em software (1 de 2)



- Qual a qualidade máxima que um produto de software pode ter?
A qualidade de um sistema ou produto é influenciada pela qualidade do processo utilizado para desenvolvê-lo e mantê-lo (CHRISSIS *et al.*, 2003, p. 5);
 - Qual a qualidade máxima que um produto de software desenvolvido pela empresa “X” pode ter?
 - Qual a qualidade mínima que um produto de software pode ter?
- Enquanto a maioria dos tratamentos de qualidade são descritos em termos do software final e performance do sistema, a prática de engenharia requer que artefatos intermediários relevantes tenham qualidade avaliada ao longo do processo de Engenharia de Software;
 - Avaliar se um artefato é relevante é negociar qualidade;

Negociação da qualidade em software (2 de 2)



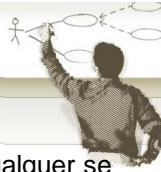
- Exemplos de artefato incluem (e não estão limitados a):
 - Uma especificação de requisitos de software para um componente de um sistema;
 - Um diagrama técnico;
 - Código fonte;
 - Documentação de testes;
 - Relatórios produzidos como resultado de tarefas de análise de qualidade;
 - Etc.
- Neste sentido, todos os processos (ou todas as Áreas de Processo) do CMMI-DEV influenciam na negociação da qualidade:
 - Se o processo é uma “receita de bolo”:
 - Deixar de adicionar um ingrediente pode comprometer o resultado;
 - Executar alguma tarefa apenas parcialmente pode comprometer o resultado.

Exemplo de qualidade máxima para um produto – a partir de processos de Engenharia e Gestão



- Se um processo de software bastante simples utilizado pela empresa “X” determina que devem ser mantidos (criados e atualizados, quando necessário) os artefatos:
 - Sob perspectiva da Engenharia:
 - Lista de Requisitos (texto em linguagem natural);
 - Casos de Uso (diagramas e especificação);
 - Diagrama de Classes de Análise;
 - Especificação de casos de teste;
 - Programas Fonte;
 - Sob perspectiva da Gestão:
 - Plano de Projeto;
 - Cronograma;
 - Plano de testes.

Exemplo de qualidade negociada para um produto – a partir de processos de Engenharia e Gestão



- Mas em um projeto específico executado para um cliente qualquer se define que devem ser mantidos (criados e atualizados, quando necessário) APENAS os artefatos:
 - Sob perspectiva da Engenharia:
 - Lista de Requisitos (texto em linguagem natural);
 - Programas Fonte;
 - Sob perspectiva da Gestão:
 - Cronograma;

→ Caracteriza-se uma negociação da qualidade

- Evidentemente, por se tratar de uma negociação a decisão não pode ser tomada de maneira unilateral (cliente e outros afetados devem estar cientes que a qualidade vai ser reduzida – em relação ao padrão – e devem compreender as implicações).

Tema qualidade abordado de forma explícita no CMMI-DEV



- Os processos de Gestão e Engenharia utilizados influenciam na qualidade do produto;
- O CMMI provê um guia para melhoria de processos – logo, melhoria da qualidade;
- Áreas de processo do modelo que mencionam explicitamente a qualidade envolvem:
 - Garantia de Qualidade de Processo e Produto (PPQA – ML-2);
 - Verificação (VER – ML-3);
 - Validação (VAL – ML-3).

PMBOK: Processos de Qualidade



- A área de conhecimento de Gerenciamento de Qualidade do Projeto (PMBOK 4a. Edição) informa que os processos de qualidade, no contexto de gerenciamento de projetos em geral:
 - Incluem todas as atividades da organização executora que determinam as responsabilidades, os objetivos e as políticas de qualidade, de modo que o projeto atenda às necessidades que motivaram sua realização;
 - Os processos do gerenciamento de qualidade envolvem:
 - Planejar a qualidade;
 - Realizar a garantia da qualidade;
 - Realizar o controle da qualidade.

Planejamento da Qualidade



- O planejamento da qualidade inclui a identificação de padrões relevantes para o projeto e a determinação de como satisfazê-los;
- Elaborado durante o planejamento do projeto:
 - O padrão de qualidade a adotar têm influência em recursos, prazos, custos;
 - O planejamento da qualidade é uma forma de negociar a qualidade;
- Sob perspectiva do CMMI-DEV: planejamento do projeto (PP, PMC, IPM).

Realizar a garantia da qualidade



- Executar garantia de qualidade envolve aplicar as atividades de qualidade planejadas, de modo que o projeto utilize todos os processos necessários (planejados) durante sua execução;
- A garantia de qualidade tem caráter consultivo;
- Sob perspectiva do CMMI-DEV: PPQA.

Realizar o controle da qualidade



- O controle da qualidade envolve monitoração dos resultados do projeto;
- O controle da qualidade deve ser realizado durante todo o projeto;
- Sob perspectiva do CMMI-DEV: VER e VAL.



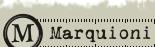
Visão geral do CMMI-DEV

www.marquioni.com.br

Porque usar um modelo como referência?

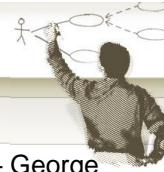


- O uso do CMMI como de referência:
 - Define uma linguagem comum;
 - Aplica *benchmarking* de qualidade (o modelo foi elaborado com base na experiência acumulada pela comunidade de software);
 - Fornece uma estrutura para priorização de ações;
 - Fornece uma estrutura para a execução de avaliações confiáveis e consistentes.



www.marquioni.com.br

Então não tem como dar errado!



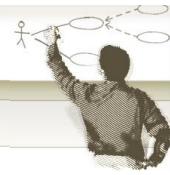
- “Todos os modelos têm problemas, mas alguns são úteis” – George Box (Engenheiro de Qualidade);
- Modelos são simplificações do mundo real;
- Modelos não são suficientemente abrangentes;
- Interpretação e adaptação a situações particulares devem estar ajustadas aos objetivos do negócio;
- Costuma-se dizer que é necessário bom senso para utilizar modelos corretamente e com visão → Alto Risco!

O que o modelo não cobre

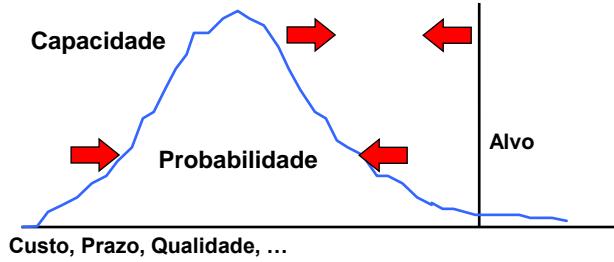


- O modelo não aborda todos os processos de software e todas as questões ligadas à qualidade;
- Questões que são abordadas indiretamente ou por consequência, incluem, por exemplo:
 - Ferramentas específicas, métodos e tecnologias;
 - Trabalho em equipe;
 - Comportamento organizacional.

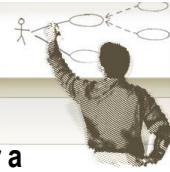
Objetivos básicos do modelo (1 de 2)



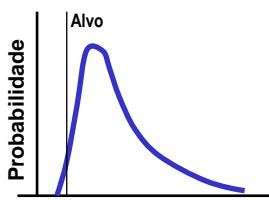
- Priorizar ações a tomar;
- Aproximar o previsto do realizado;
- Diminuir a dispersão (acertar mais vezes).



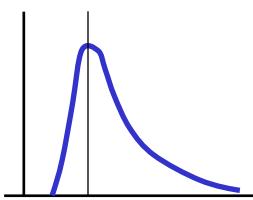
Objetivos básicos do modelo (2 de 2)



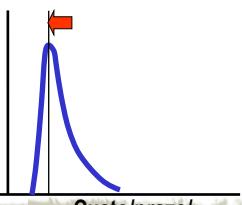
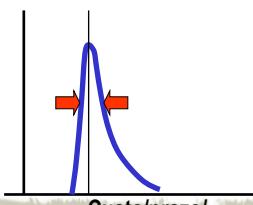
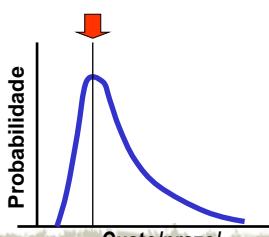
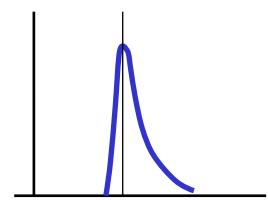
Melhorar a
Previsibilidade



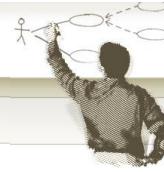
Melhorar o
Controle



Melhorar a
Eficiência



Histórico CMM



- CMM – *Capability Maturity Model*
 - Meados anos 1980;
 - Realidade do DoD (Departamento de Defesa) Norte Americano: projetos de software:
 - Atrasados
 - Com custo acima do orçado;
 - Solicitada à *Carnegie Mellon University* a formação de um grupo de excelência para:
 - Definir o que fazer para produzir software com qualidade;
 - Definir uma priorização de ações (níveis de maturidade);
 - Fundado o SEI (*Software Engineering Institute*).

Resultado do trabalho do grupo de excelência



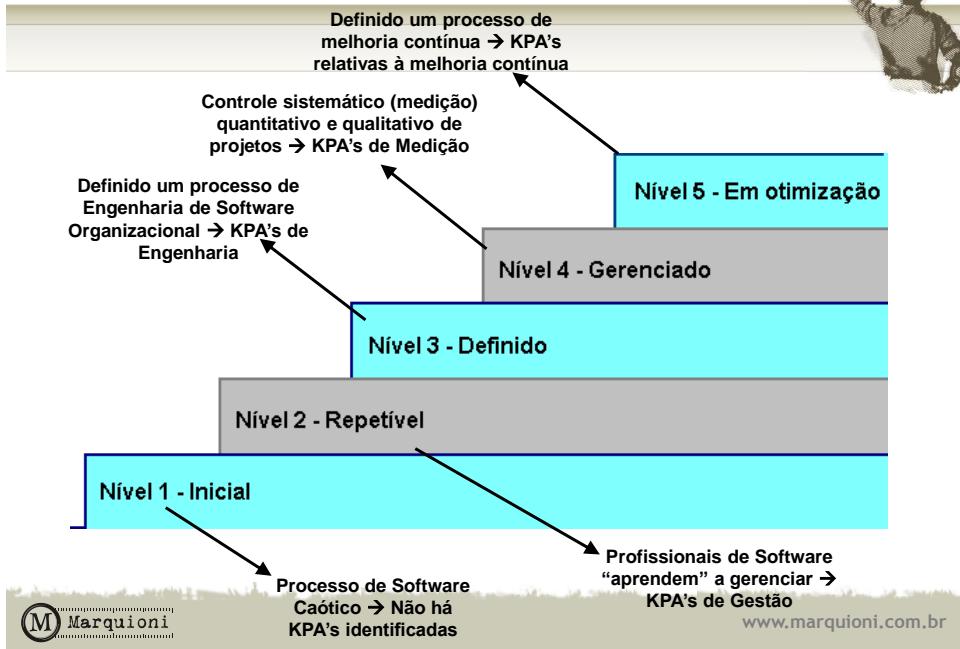
- Lista de atividades que deveriam ser executadas para produzir software com qualidade:
 - Realizar revisão por pares;
 - Utilizar um processo de engenharia de software;
 - Criar e *atualizar* cronograma;
 - Criar e *atualizar* plano de projeto;
 - Controlar versões;
 - Controlar mudanças;
 - Estimar custo, esforço, duração;
 - Gerenciar os requisitos;
 - Realizar auditorias no uso dos processos;
 - Coletar métricas;
 - Definir um programa de melhoria contínua;
 - ...
- A lista ficou extensa; foi criada uma estrutura para priorização de ações...

Priorização estabelecida - CMM

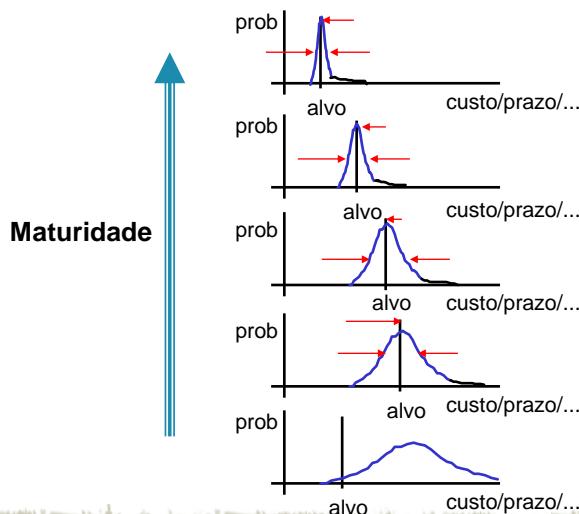


- Atividades agrupadas de acordo com processos relativos a:
 - Gestão de projetos;
 - Engenharia de Software;
 - Medição;
 - Melhoria contínua;
- Os agrupamentos de processos:
 - Foram nomeados Áreas Chave de Processo (KPA – Key Process Area);
 - As KPA's foram distribuídas em níveis, constituindo uma sugestão de priorização para os processos a executar...

Priorização estabelecida por níveis - CMM



Objetivos do modelo x Maturidade Organizacional

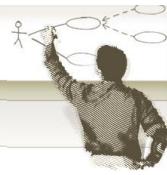


Histórico CMMI



- CMMI 1.1 – *Capability Maturity Model Integration*;
 - Início anos 2000;
 - Fusão de vários CMM;
- Principais alterações CMM → CMMI 1.1;
 - Termos: KPA → PA (*Process Area* – Área de Processo);
 - Nomes de PA's;
 - Esclarecimentos em relação a nomes de KPA's (PA's);
 - Maior ênfase aos processos de requisitos;
 - Maior ênfase a alguns aspectos gerenciais (riscos, p. ex.);
 - Maior ênfase aos processos de medição (PA nível 2);
 - Simplificada a estrutura de organização interna de cada PA;
- A partir de 2007: CMMI 1.2:
 - Redução da complexidade e tamanho do modelo (principalmente estrutura de organização interna de cada PA);
 - Aumento de clareza.

E porque CMMI-DEV?

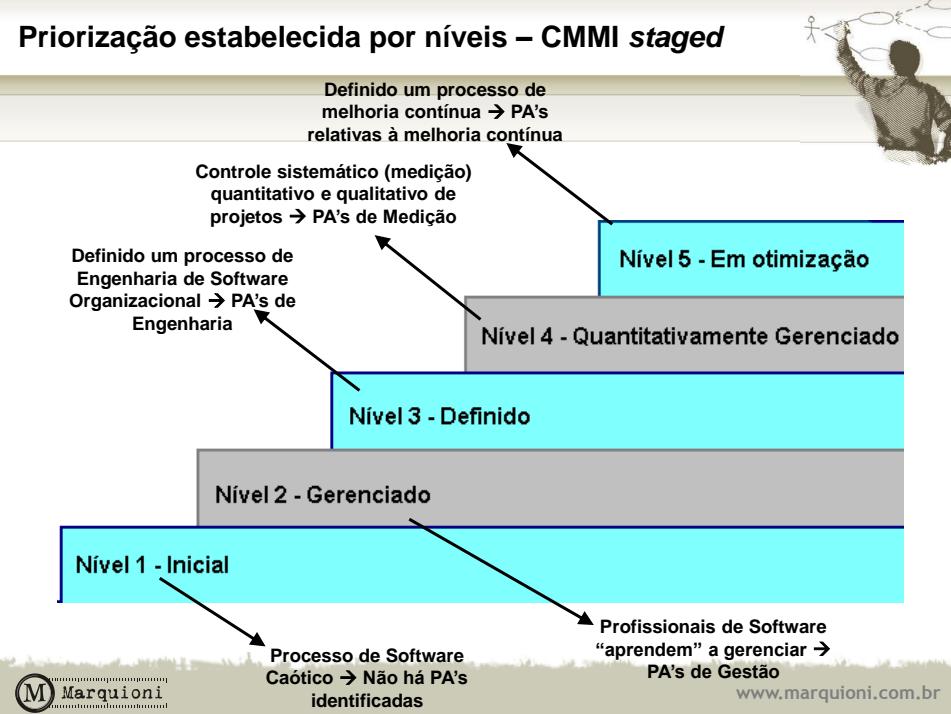


- **Constelação:**
 - Uma constelação é um subconjunto da Suíte de Produtos CMMI para melhoria de uma área de interesse em particular;
 - Constelações CMMI:
 - Desenvolvimento;
 - Aquisição;
 - Serviços;
 - CMMI-DEV = CMMI-*Development*:
 - Objeto da constelação: desenvolvimento/manutenção de software;
 - Outras constelações CMMI:
 - CMMI-Acquisition: para compra de pacotes (2007);
 - CMMI-Services: similar ao *ti ITIL Service delivery and support* (2009);
 - Todas as referências neste material ao CMMI é relativo à constelação CMMI-DEV.

Histórico CMMI – Parte 2



- A partir de 2010: CMMI 1.3;
- Principais alterações CMMI 1.2 → CMMI 1.3:
 - ‘Produtos de trabalho típicos’ passam a ser referenciados como ‘Exemplos de produtos de trabalho’;
 - Simplificação por redução de amplificações e adições;
 - PA REQM ‘deslocada’ da categoria de Engenharia para Gestão;
 - Inclusão de tratamentos relativos aos métodos Ágeis;
 - Metas e práticas genéricas ‘concentradas’ em um único local, facilitando o acesso;
 - Aumento na clareza de termos (por exemplo GP 2.6 renomeada: de ‘Gerenciar configurações’ para ‘Controlar produtos de trabalho’);
 - Melhoria na especificação dos níveis de maturidade mais altos, visando minimizar riscos de compreensões subjetivas.



Áreas de Processo

- Agrupamento de práticas relacionadas em uma área que, quando implementadas coletivamente, satisfazem um conjunto de metas consideradas importantes para a melhoria da referida área;
- As áreas de processo são organizadas por:
 - Nível de maturidade na representação *por estágios*;
 - Categorias de áreas de processo (engenharia, gestão de projeto, gestão de processo e suporte) na representação *contínua*;
- Há 22 áreas de processo no CMMI v 1.3.

Organização das Áreas de Processo (1 de 2)



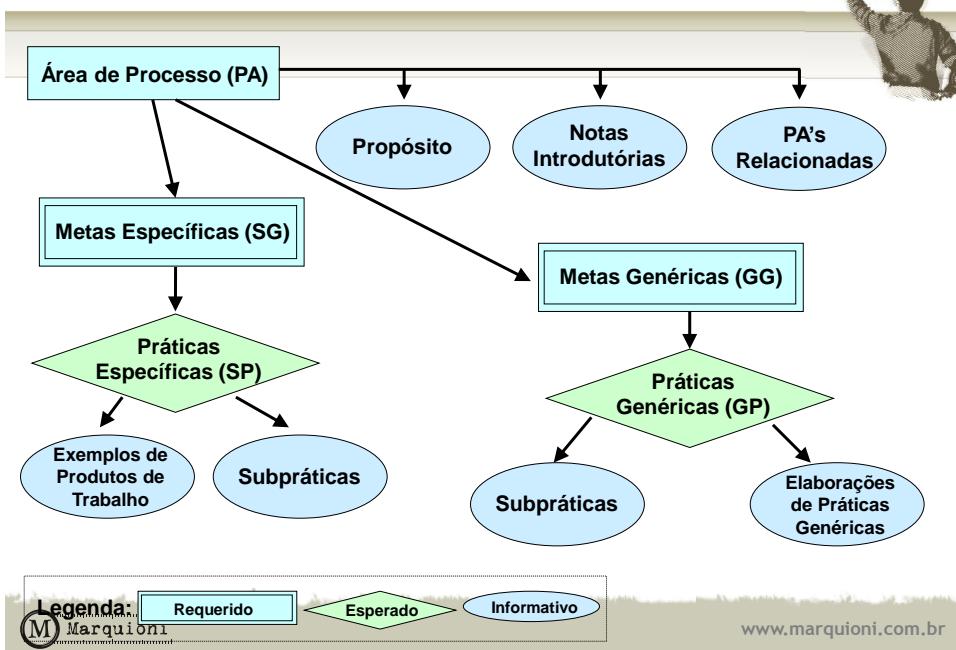
- As 22 áreas de processo são organizadas no CMMI v. 1.3 de acordo com a ordem alfabética do acrônimo da PA (em inglês):
 - CAR – Análise e Resolução de Causas [de problemas];
 - CM – Gestão de Configuração;
 - DAR – Análise de Decisão e Resolução;
 - IPM – Gestão Integrada de Projetos;
 - MA – Medição e Análise;
 - OPD – Definição do Processo Organizacional;
 - OPF – Foco no Processo Organizacional;
 - OPM – Gestão da Performance Organizacional;
 - OPP – Performance do Processo Organizacional;
 - OT – [Programa de] Treinamento Organizacional;

Organização das Áreas de Processo (2 de 2)



- Continuação:
 - PI – Integração do Produto;
 - PMC – Monitoração e Controle de Projeto;
 - PP – Planejamento de Projeto;
 - PPQA – Garantia de Qualidade de Processo e Produto;
 - QPM – Gestão Quantitativa de Projetos;
 - RD – Desenvolvimento de Requisitos;
 - REQM – Gestão de Requisitos;
 - RSKM – Gestão de Riscos;
 - SAM – Gestão de Acordos com Fornecedores;
 - TS – Solução Técnica;
 - VAL – Validação;
 - VER – Verificação.

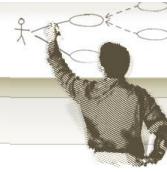
Estrutura Áreas de Processo CMMI v. 1.3



Requerido, Esperado, Informativo

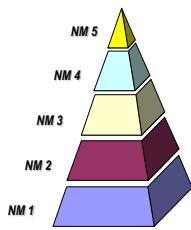
- **Requerido:** descreve o que a organização obrigatoriamente deve alcançar para satisfazer uma área de processo;
- **Esperado:** descreve o que a organização irá tipicamente implementar para alcançar um componente requerido. Funciona como guia para aqueles que implementam as melhorias e para aqueles que executam as avaliações;
- **Informativo:** provê detalhes que auxiliam a organização a pensar em como abordar os componentes requeridos e esperados.

Estrutura de representação do modelo



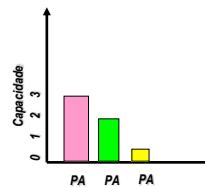
- Há 2 tipos de representação nos modelos CMMI:
 - Por estágios (*estagiada, staged*) – 5 níveis (1 a 5);
 - Contínua (*continuous*) – 4 níveis (0 a 3).

POR ESTÁGIOS



Organização

CONTÍNUA



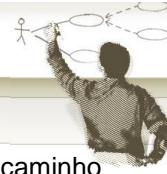
Processo

Representações do CMMI (2 de 2)



- Ambas representações provêm:
 - Formas de implementação de melhorias de processo para alcançar metas de negócio;
 - Essencialmente o mesmo conteúdo e usam os mesmos componentes do modelo - apenas estão organizadas de formas diferentes.

Entendendo o conceito de Nível (1 de 2)



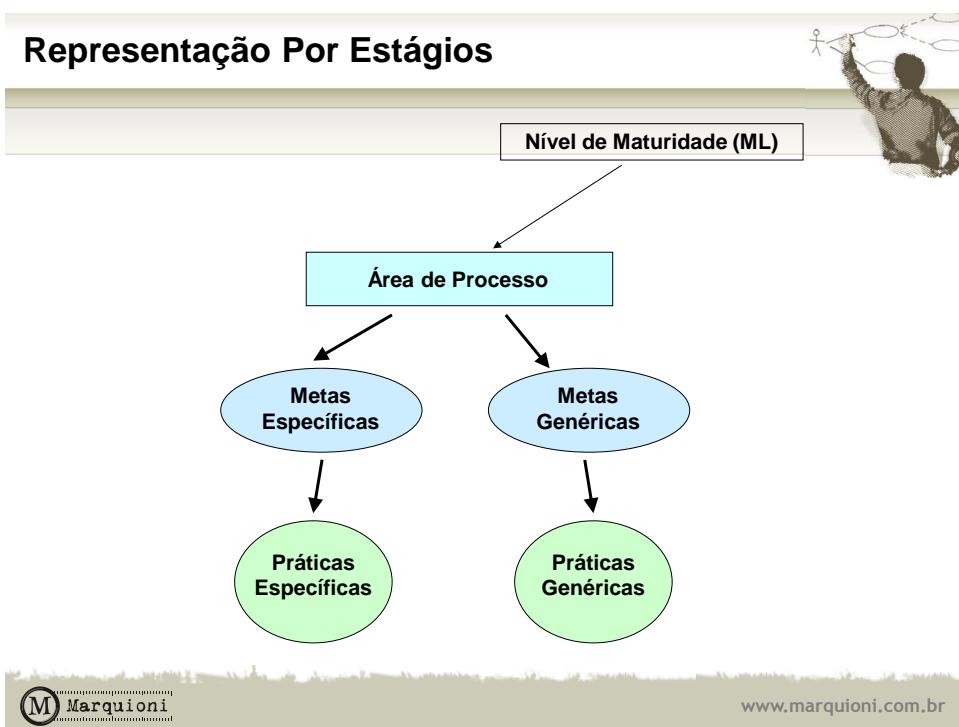
- É utilizado o conceito de nível no CMMI para descrever um caminho evolutivo para uma organização que deseja melhorar os processos que utiliza para desenvolver e manter seus produtos e serviços;
- O CMMI suporta dois caminhos de melhoria:
 - **Por nível de capacidade:** termo aplicável para melhoria de processos quando utilizada a representação contínua. Este caminho de melhoria habilita melhoria incremental para uma área de processo individualmente;
 - **Por nível de maturidade:** termo aplicável para melhoria de processos quando utilizada a representação por estágios. Este caminho de melhoria habilita melhoria incremental para um conjunto predefinido de áreas de processo relacionadas.

Entendendo o conceito de Nível (2 de 2)



- Apesar da variação de PA's por caminhos de melhoria, o conceito de nível é o mesmo;
- Níveis caracterizam melhorias entre etapas através do uso de informações para determinar e gerenciar estas melhorias, que são necessárias para atingir os objetivos de negócio da organização;
- Para alcançar um nível em particular, a organização deve satisfazer todas as metas apropriadas da área de processo (ou do conjunto de áreas de processos) alvo para a melhoria.

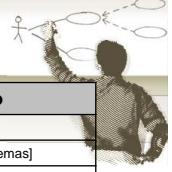
Representação Por Estágios



Níveis de maturidade não podem ser omitidos

- Cada nível de maturidade provê os fundamentos necessários para implementação efetiva dos processos de nível seguinte:
 - Processos de nível mais elevado têm chance maior de sucesso com a disciplina provida pelos níveis inferiores;
 - O efeito de inovações de maturidade maiores são mais facilmente mensuráveis;
- Processos de nível de maturidade superior podem ser executados por organizações de níveis de maturidade inferior com o riscos de não serem aplicadas de forma consistente em situações de crise.

Representação Por Estágios: PA's por Nível de Maturidade



Produtividade e Qualidade	Nível		Foco	Áreas de Processo
5	Em Otimização	Melhoria contínua do processo		Gestão da Performance Organizacional Análise e Resolução de Causas [de problemas]
4	Quantitativamente Gerenciado	Gerenciamento Quantitativo		Performance do Processo Organizacional Gestão Quantitativa de Projetos
3	Definido	Padronização Organizacional de Processos		Desenvolvimento de Requisitos Solução Técnica Integração do Produto Verificação Validação Foco no Processo Organizacional Definição do Processo Organizacional [Programa de] Treinamento Organizacional Gestão Integrada de Projetos Gestão de Riscos Análise de Decisão e Resolução
2	Gerenciado	Gerenciamento Básico de Projetos		Gestão de Requisitos Planejamento de Projeto Monitoração e Controle de Projeto Gestão de Acordos com Fornecedores Medição e Análise Garantia de Qualidade de Processo e Produto Gestão de Configuração
1	Início	n/a		n/a

www.marquioni.com.br

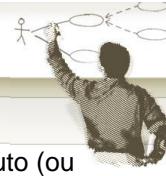
Risco de
 Marquioni



Visão geral das PA's abordadas

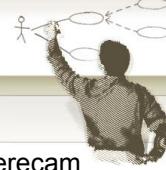
www.marquioni.com.br

VAL x VER x PPQA



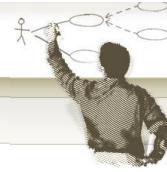
- O propósito da Validação (VAL) é demonstrar que um produto (ou componente de produto) atende ao uso desejado, quando colocado no ambiente pretendido;
 - VAL (Validação): avalia se o produto está adequado ao uso;
- O propósito da Verificação (VER) é garantir que os produtos de trabalho selecionados atendem aos requisitos correspondentes especificados;
 - VER (Verificação): avalia se o produto está tecnicamente aderente à especificação;
- O propósito da Garantia de Qualidade de Processo e Produto (PPQA) é prover a equipe de trabalho e a gerência com informações objetivas a respeito dos processos e dos produtos de trabalho associados;
 - PPQA (Garantia da Qualidade): avalia se o processo customizado para o projeto/produto tem sido utilizado adequadamente (conforme o planejado/customizado);

Verificação x Validação (1 de 2)



- As PA's de Verificação e Validação são similares, mas endereçam aspectos distintos:
 - **Validação** demonstra que o produto, da forma como entregue, atende plenamente ao uso desejado (“você construiu a coisa certa”: o produto faz o que o usuário deseja);
 - Verificação endereça se o produto de trabalho foi construído em conformidade com a especificação do requisito (“você construiu a coisa corretamente”: o produto foi construído de acordo com o especificado).
- As atividades de validação usam abordagens similares à verificação (testes, análise, inspeção, demonstração ou simulação).

Verificação x Validação (2 de 2)



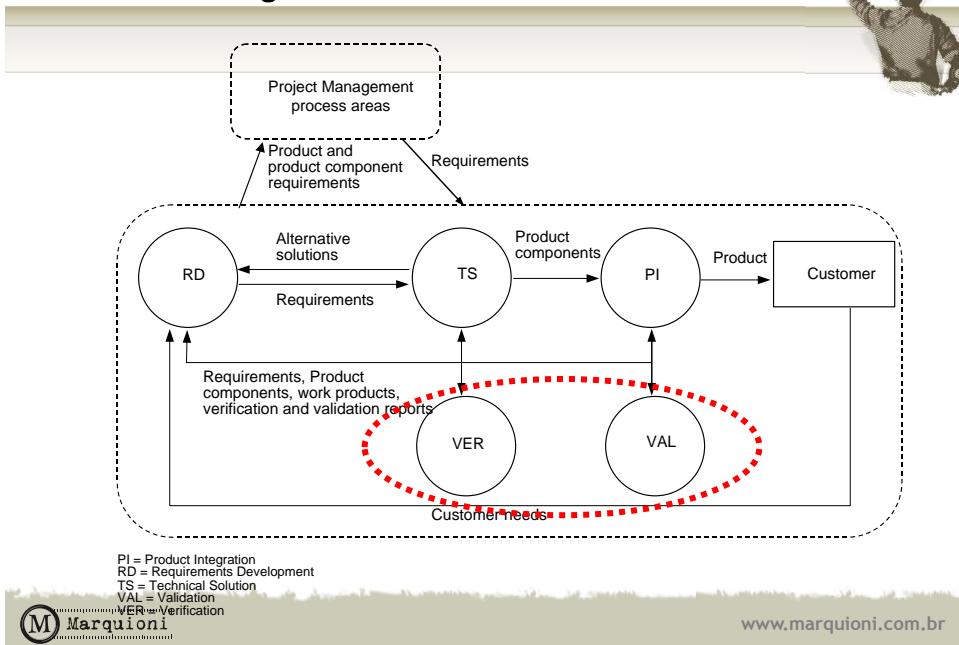
- Requisito: construir um edifício comercial
 - **Validação:** é possível montar escritórios no edifício construído;
 - Verificação: as salas do edifício contém a metragem definida na planta.
- Requisito: construir um software de controle de estoque
 - **Validação:** é possível movimentar itens de estoque no software;
 - Verificação: a regra de negócio de estoque mínimo (definida na lista de requisitos) está corretamente implementada.
- Requisito: construir um software para cadastro de clientes e endereços respectivos
 - **Validação:** é possível cadastrar clientes e endereços no software;
 - Verificação: a integridade referencial definida (nos modelos de dados conceitual e lógico) está corretamente implementada.

PPQA nos exemplos anteriores

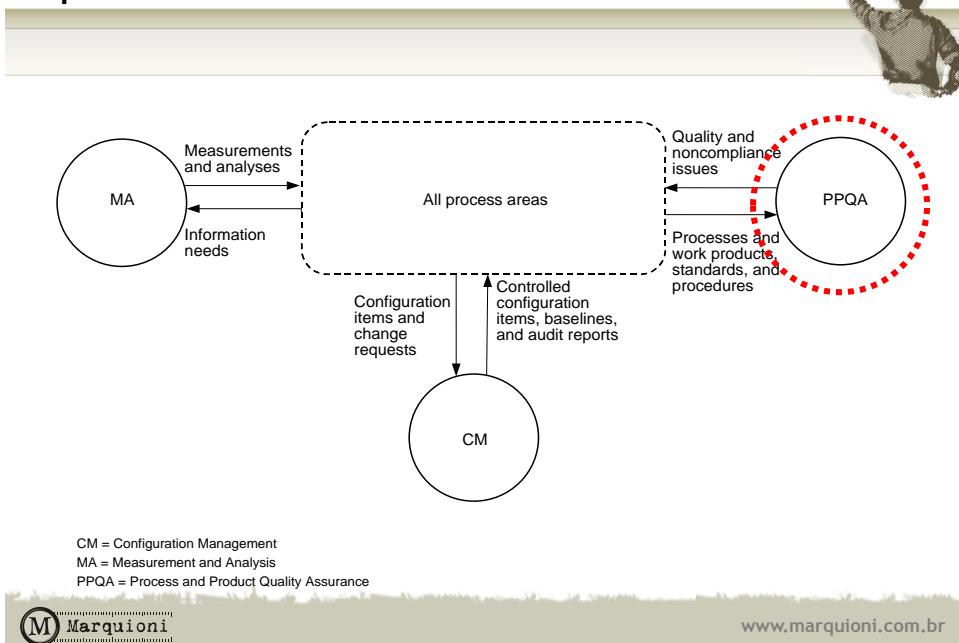


- Auditar se todos os artefatos esperados foram efetivamente elaborados (se a qualidade negociada foi aplicada):
 - Para o edifício comercial:
 - Planta Baixa, Planta Hidráulica, Planta Elétrica etc.
 - Para os softwares:
 - Lista de Requisitos, casos de uso, programas fonte etc;
- Cuidado! Auditoria neste contexto possui caráter consultivo, e não necessariamente punitivo.

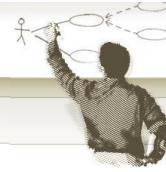
Contextualização Geral – VER e VAL: Grupo de Processos de Engenharia



Contextualização Geral – PPQA: Grupo de Processos de Suporte



Referências Bibliográficas



- BEZERRA, Eduardo. *Princípios de Análise e Projeto de Sistemas com UML*. Rio de Janeiro: Campus, 2007;
- CMMI-DEV. *CMMI for Development Version 1.3: Improving processes for developing better products and services*. Hanscom, 2010. Disponível em: <<http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm>>. Acesso em: 25/07/2011.
- CRISSIS, Mary Beth; KONRAD, Mike; SHRUM, Sandy. *CMMI Guidelines for process integration and produc improvement*. Boston: Addison-Wesley, 2003.
- FUNDAÇÃO BIBLIOTECA NACIONAL (Rio de Janeiro). C.E. Marquioni. *Treinamento na área de processo Verificação*. EDA. BR, No. Registro 424.866, Livro 795, Folha 26, 18/02/2008.
- FUNDAÇÃO BIBLIOTECA NACIONAL (Rio de Janeiro). C.E. Marquioni. *Treinamento na área de processo Validação*. EDA. BR, No. Registro 414.223, Livro 773, Folha 383, 26/10/2007.

Referências Bibliográficas



- FUNDAÇÃO BIBLIOTECA NACIONAL (Rio de Janeiro). C.E. Marquioni. *Treinamento na área de processo Garantia de Qualidade de Processo e Produto*. EDA. BR, No. Registro 414.221, Livro 773, Folha 381, 26/10/2007.
- INTRODUCTION to CMMI® Version 1.2. Training workbook. © 2006 by Carnegie Mellon University.
- PMBOK. *A Guide to the Project Management Body of Knowledge: 4th Edition*. Pennsylvania: PMI Publications, 2008.
- PRÁTICA. Processo de Engenharia e Gestão de Software. Marquioni, C.E., 2006.
- PRESSMAN, Roger. *Software Engineering – A Practitioner's Approach – European Adaptation*. London: McGraw Hill International Limited, 2000.