



1

ENGENHARIA DE SOFTWARE

Ciência da Computação - UniBH

Criado por Paulo Henrique Ladeira

CONTEXTUALIZAÇÃO DA DISCIPLINA

- Informações gerais:

- Engenharia de software (CIC4BN-ESA).
- Carga horária: 80 horas.
- Professor: Paulo Henrique Ladeira.
- E-mail: paulo.ladeira@prof.unibh.br
- Horário: Segundas (19:00 as 20:35) e quartas (19:00 as 20:35)
- Bibliografia básica:
 - Engenharia de software (8th ou 9th Edição) - Ian Sommerville.
 - Engenharia de software (5th ou 6th Edição) - Roger S. Pressman;

CONTEXTUALIZAÇÃO DA DISCIPLINA

- Ementa da disciplina:

- Fundamentos de engenharia de software;
- Conceitos de gestão de projetos (Escopo, Prazo, Custo, Qualidade, Risco, Pessoas, Comunicação, etc);
- Gestão de projetos com metodologias ágeis (SCRUM);
- Processos de software (Modelos de maturidade, RUP);
- Métricas de software (Técnicas de estimativa);
- Técnicas de teste.

ENGENHARIA

- Arte de aplicar conhecimentos
 - científicos
 - empíricos
 - certas **habilitações específicas**
- à criação de estruturas, dispositivos e processos que se utilizam para converter recursos naturais em formas adequadas ao atendimento de necessidades humanas.

[Aurélio eletrônico]

ELEMENTOS DA ENGENHARIA

- Conhecimentos científicos:
 - parte dos métodos da Engenharia de Software provém da Ciência da Computação.
- Conhecimentos empíricos:
 - parte dos métodos da Engenharia de Software provém da experiência prática.
- Habilidades específicas
 - a Engenharia de Software possui um conjunto de habilidades específicas, ou disciplinas.

ELEMENTOS DA ENGENHARIA

- Arte:
 - capacidade de pôr em prática uma ideia para dominar a matéria;
- Na Engenharia de Software:
 - máquinas de processamento da informação configuradas e programadas.
- Atendimento das necessidades humanas:
 - satisfação de necessidades é gerar algo que tenha valor para alguém
 - Engenharia de Software procura gerar valor por meio dos recursos de processamento de informação

INTRODUÇÃO A ENGENHARIA DE SOFTWARE

- A Engenharia de software trata-se de aspectos relacionados ao estabelecimento de **processos, métodos, técnicas, ferramentas e ambientes de suporte** ao desenvolvimento de software.
- Quando falamos de Engenharia de Software não falamos exclusivamente do programa. Inclui-se:
 - Toda a documentação associada;
 - Dados de configurações necessários;
 - Processos utilizados durante o desenvolvimento;
 - Etc.

INTRODUÇÃO A ENGENHARIA DE SOFTWARE

- Existem **processos, métodos, técnicas e ferramentas** universais para a Engenharia de Software?
 - Não. Diferentes tipos de softwares exigem abordagens diferentes.
- O que os softwares abaixo tem em comum?
 - Software corporativo;
 - Jogo;
 - Aplicativo para celular;
 - Software BB;
 - Software de uma UTI.
- Todos dependem da Engenharia de Software, embora não necessitem da mesma técnica.

INTRODUÇÃO A ENGENHARIA DE SOFTWARE

- Quais os limites para o desenvolvimento de um software?
 - Nenhum.
- Isso simplifica a engenharia do software, dado que hoje, não existem limitações naturais para um software.
- Porém, dada a ausência de restrição físicas, os softwares tendem a se tornar cada vez mais complexos e difíceis de serem desenvolvidos.

INTRODUÇÃO A ENGENHARIA DE SOFTWARE

- Existem basicamente dois tipos de softwares:
 - Informal
 - Planilhas do excel para simplificar o trabalho;
 - Cientistas para processar os dados;
 - Pessoas comum por *hobby* (Ex: Controle de finanças).
 - Profissional  Engenharia de Software
 - Contêm usuários além do próprio desenvolvedor;
 - Criado por equipes (conhecimento não centralizado);
 - Mantido e alterado durante sua existência.

INTRODUÇÃO A ENGENHARIA DE SOFTWARE

- O que é software?

- Programa de computador **mais** a documentação associada. Pode ser desenvolvida para um cliente específico ou para o mercado em geral (produto).

- Quais os principais atributos de um bom software?

- Prover a funcionalidade e o desempenho esperado pelo usuário;
- Confiável;
- Fácil manutenção e utilização.

QUAIS OS PRINCIPAIS ATRIBUTOS DE UM SOFTWARE CONFIÁVEL?

- Manutenabilidade
 - Software deve ser fácil de se manter e atender as necessidades dos clientes (mudança é inevitável).
- Confiança, proteção e segurança
 - Não pode causar prejuízo físico ou econômico em caso de falha.
- Eficiência
 - Software precisa ser rápido, simples e eficiente em relação aos recursos do sistema.
- Usabilidade
 - Software precisa ser fácil de usar de forma intuitiva.

O QUE É ENGENHARIA DE SOFTWARE?

- É uma disciplina de engenharia que se preocupa com todos os aspectos da produção do software.
- Engenheiros de software **devem** - dependendo do problema a ser resolvido, das restrições de desenvolvimento e dos recursos disponíveis - **adotar uma abordagem sistemática e organizada para seu trabalho**, além de usar ferramentas e técnicas apropriadas.

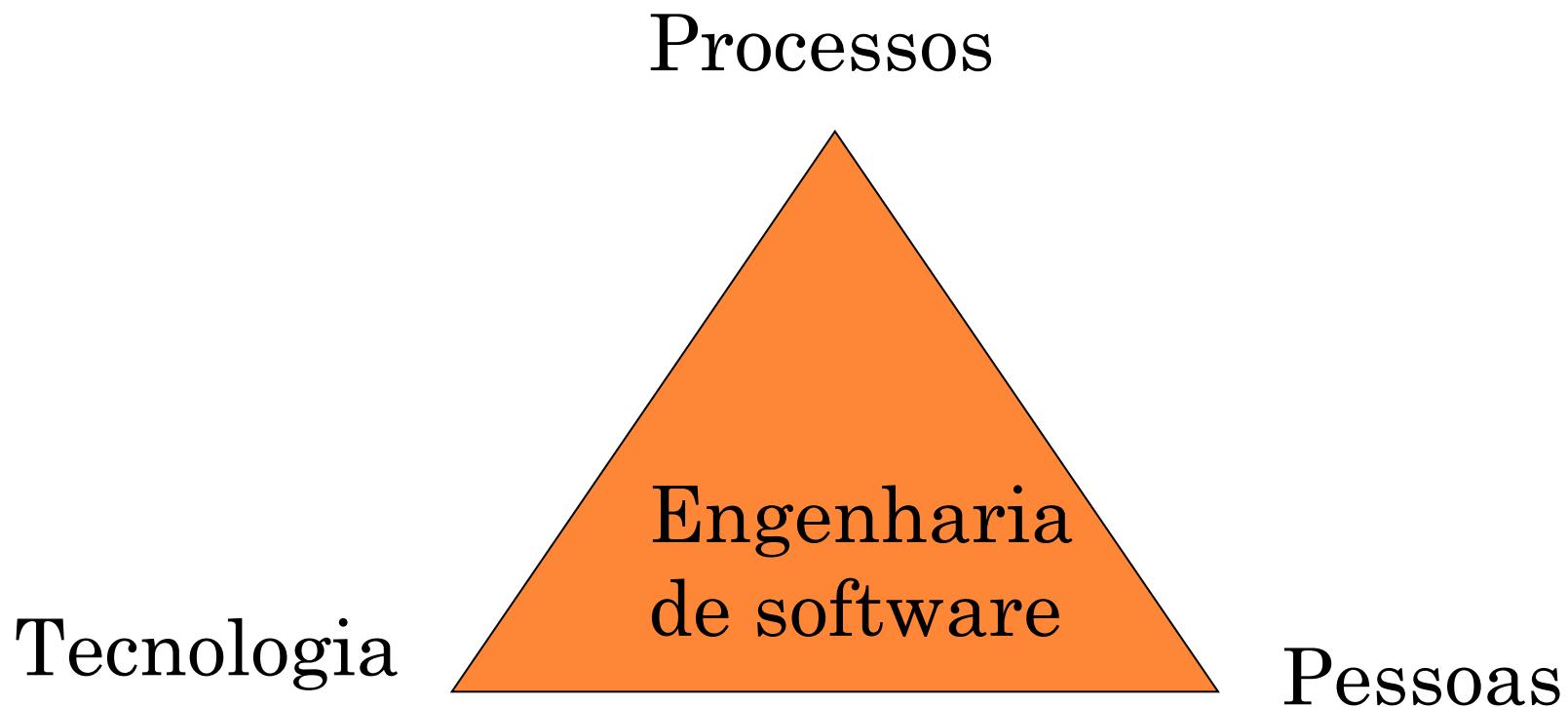
INTRODUÇÃO A ENGENHARIA DE SOFTWARE

- Qual a diferença entre engenharia de software e ciência da computação?
 - Ciência da computação foca na **teoria** e nos **conceitos**;
 - Engenharia de software preocupa-se com o lado **prático** do desenvolvimento de entrega de softwares úteis.
- Qual a diferença entre engenharia de software e de sistemas?
 - Engenharia de sistema se preocupa com **todos os aspectos** do desenvolvimento de sistemas (hardware, software e processo).
 - Engenharia de software é **uma parte** específica desse processo.

INTRODUÇÃO A ENGENHARIA DE SOFTWARE

- Existem dois tipos de produtos de software:
 - Genéricos;
 - Produzido por uma organização, responsável por manter a especificação e vendido para quem tiver interesse em comprar;
 - Ex: Pacote office, dreamweaver, coreldraw, etc.
 - Sob encomenda.
 - Sistema encomendado por um cliente específico, onde a contratante é responsável pela especificação.
- Porém, está cada vez mais comum um meio termo entre esses dois tipos.
 - Exemplo: Sistemas ERP (Sistema integrado de gestão empresarial), como o SAP

OS TRÊS PILARES DA ENGENHARIA DE SOFTWARE



PESSOAS

- Cliente
- Usuário final
- Equipe do projeto
 - Analistas
 - Projetistas
 - Desenvolvedores
 - Testadores

TECNOLOGIA

- Linguagens de programação
 - Java, C, DotNet, etc.
- Ferramentas do projeto de desenvolvimento
 - Requisitos (Word, etc), Análise (Rose, ArgoUML, etc), projeto (Together – Modelagem Visual, etc), implementação (Eclipse, JBuilder, etc), testes (Robot, Junit, Selenium, Functional Tester, etc)
 - BPMS
- Banco de dados
 - Oracle, MySQL, PostgreSQL, etc

TECNOLOGIA

- Pode viabilizar o trabalho que deve ser executado
- Pode tornar o trabalho mais preciso, produtivo e eficiente
 - Cuidado para que não engesse, ou gere qualquer problema de produtividade

PROCESSOS

- Conjunto de passos parcialmente ordenados:
 - Atividades, métodos, práticas e transformações;
 - Usados para atingir uma meta;
- Meta geralmente associada a um ou mais resultados concretos finais:
 - Produtos da execução do processo

CARACTERÍSTICAS DO DESENVOLVIMENTO DE SOFTWARE

- O que se observa nas organizações atualmente?
- Há divisão de responsabilidades?
 - Requisitos, Análise, Projeto, Implementação e Testes?
- Há diferenças dependendo do porte da organização?

PROBLEMAS COMUNS NO DESENVOLVIMENTO DE SOFTWARE

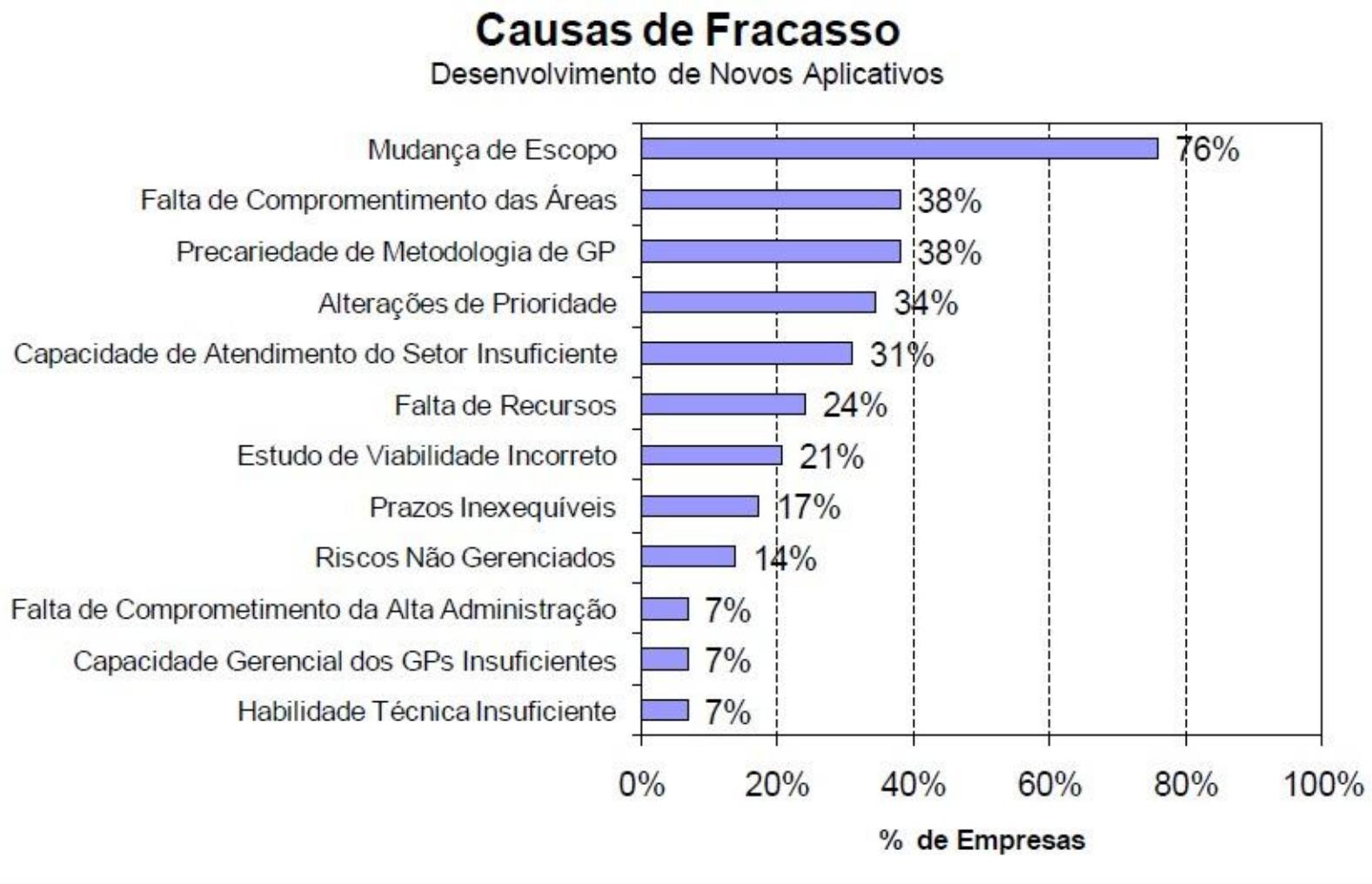
- Atrasos
- Custos extrapolados
- Requisitos não atendidos
- Produtos de baixa qualidade: lentos, com defeitos, difíceis de usar...
- Usuários insatisfeitos

Fracasso do projeto!

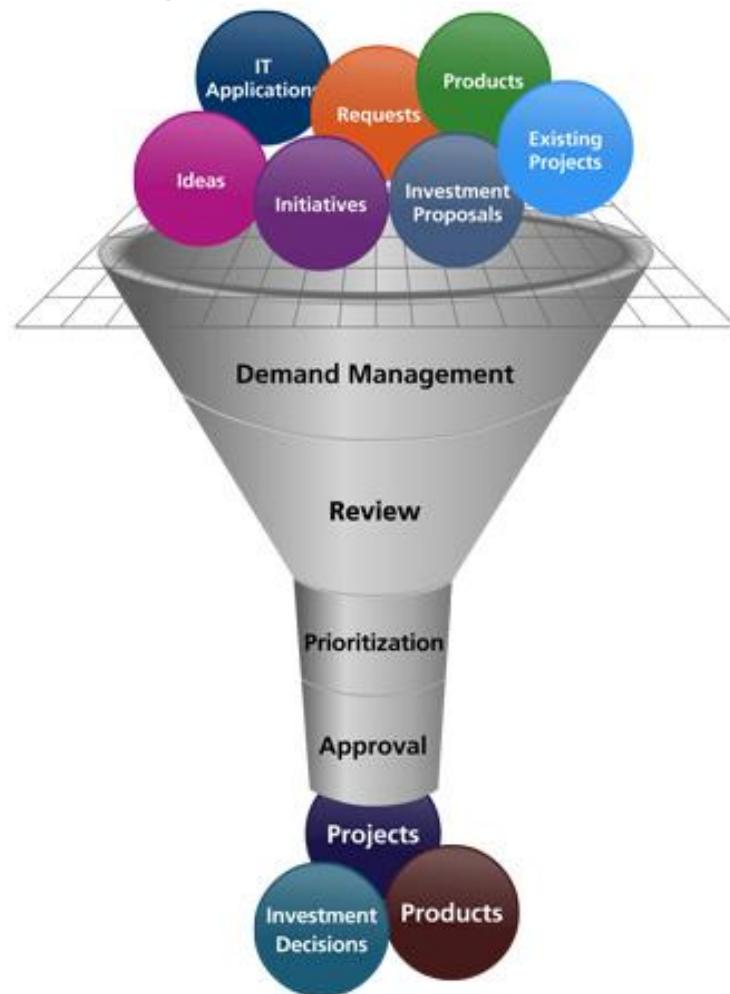
CAUSAS DOS PROBLEMAS NO DESENVOLVIMENTO DE SOFTWARE

- Estimativas de custo e prazo imprecisas
- Comunicação insuficiente entre cliente e fornecedor
- Inexperiência do gerente do projeto
- Ferramentas de automação do processo de desenvolvimento inadequadas
- Entendimento ou implantação inadequada das práticas sugeridas da literatura

CAUSAS DE FRACASSOS DE UM PROJETO

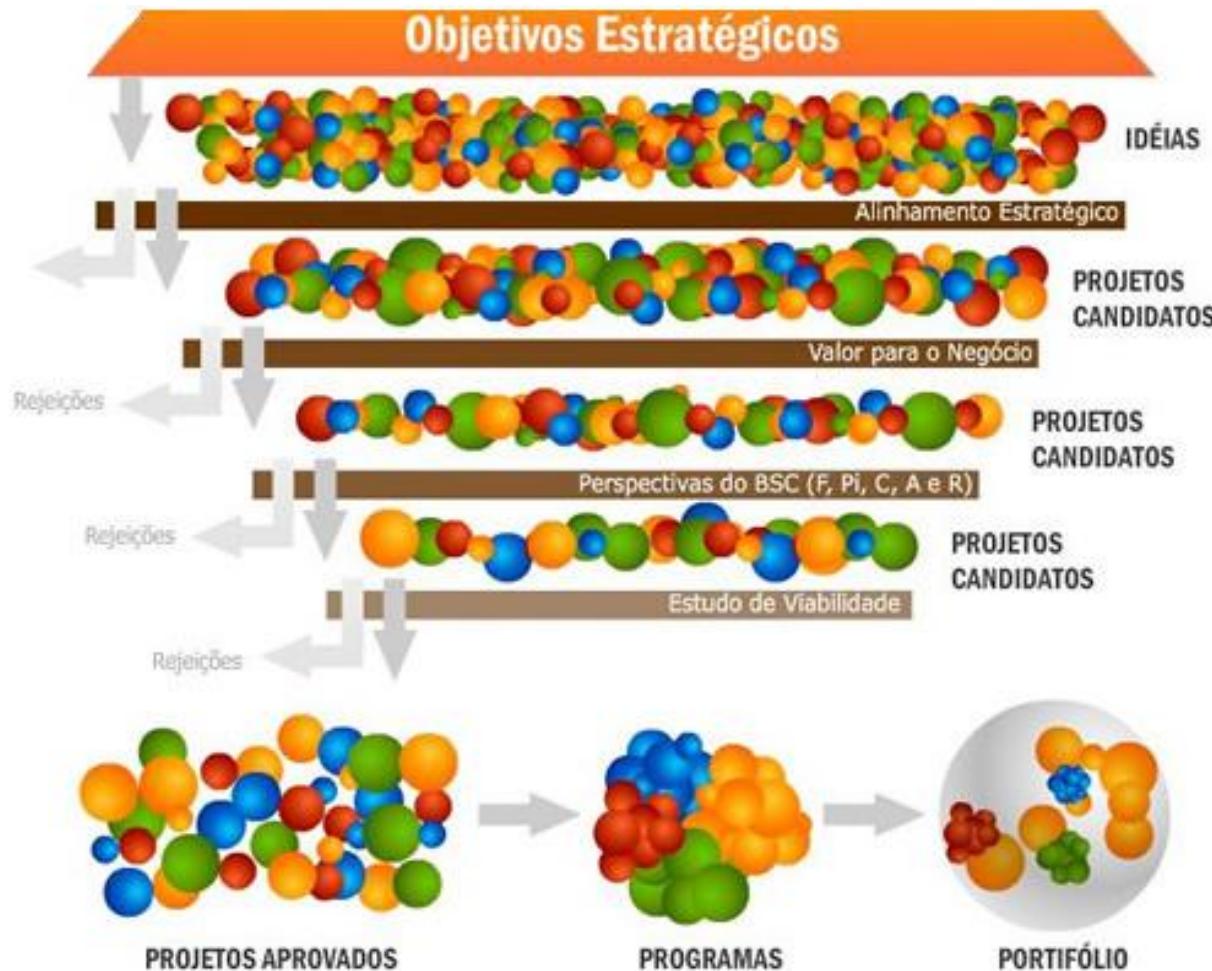


O CICLO NA ORGANIZAÇÃO



- *Fonte:* <http://www.softexpert.com.br/gestao-portfolio-projetos.php>

O CICLO NA ORGANIZAÇÃO



- Fonte: <http://evertongomede.blogspot.com.br/2010/11/01/archive.html>

O CICLO NA ORGANIZAÇÃO



Criado por Paulo Henrique Ladeira

PLANEJAMENTO ESTRATÉGICO

- A gestão estratégica trata em primeiro lugar da **formulação de estratégias que determinem rumos ou formas de atingir objetivos.**
- Essas estratégias são geralmente reunidas e descritas em um plano estratégico, que, por sua vez, é concebido didaticamente a partir de uma análise de cenários, culminando com a elaboração de uma matriz que elucide ameaças e oportunidades, sob os pontos de vista interno e externo à organização.

PLANEJAMENTO ESTRATÉGICO

- O plano estratégico será consubstanciado, então, num instrumento esclarecedor quanto:
 - à **missão** - para que servimos, qual é nossa razão de ser;
 - à **visão** - onde queremos chegar como instituição;
 - aos **valores** - quais são nossas premissas quanto às atitudes para alcançar nossa visão;
 - à **estratégia** - como faremos para alcançar nossa visão e
 - aos **desdobramentos da estratégia** - as grandes ações que precisamos conduzir e que comporão a estratégia, isto é, os *objetivos estratégicos*.

FERRAMENTAS E TÉCNICAS

- Análise SWOT
- Matriz BCG
- Plano de negócio

ANÁLISE SWOT

- Objetivo:

- Efetuar uma síntese das análises internas e externas;
- Identificar elementos chave para a gestão da empresa, o que implica estabelecer prioridades de atuação;
- Preparar opções estratégicas: Riscos/Problemas a resolver.

SWOT

Microambiente	
Forças (Strengths) <ul style="list-style-type: none">• Força 1• Força 2	Fraquezas (Weakness) <ul style="list-style-type: none">• Fraqueza 1• Fraqueza 2
Macroambiente	
Oportunidades (Opportunities) <ul style="list-style-type: none">• Oportunidade 1• Oportunidade 2	Ameaças (Threats) <ul style="list-style-type: none">• Ameaça 1• Ameaça 2

SWOT - AMBEV

Pontos Fortes:

- Inovação
- Adaptação ao público alvo
- Participação de mercado
- Recursos para comunicação
- Aceitação da marca

Fragilidades:

- Identificar produto "bola da vez" dentro do portifólio
- Mudança de metas internas
- Tecnologia interna obsoleta
- Gerenciamento de preço
- Não retenção de talentos

SWOT

Oportunidades:

- Aumento do consumo
- Crescimento de usuários de internet
- Captação de eventos com grande população jovem
- Estacionalidade do mercado (aumento de consumo no verão)
- Análises científicas apontando "benefícios" do consumo da cerveja

Ameaças:

- Geração saude;
- Ganho de mercado das concorrentes
- Marketing "agressivo" da concorrência
- Políticas governamentais para controle de consumo de bebidas
- Carga tributária aplicada para bebidas

FORÇAS

- Ganho no recebimento da matéria prima.
- Garantia de qualidade do produto final.
- Garantia de qualidade da matéria prima.
- Custo baixo do produto.
- Escalabilidade do processo produtivo.
- Melhoria sócioambiental.
- Promulgação das leis municipais.
- Aumento da renda dos transportadores.



STRENGTHS



WEAKNESSES

FRAQUEZAS

- Presença de contaminações em RCD
- Preconceito quanto ao uso de agregado reciclado.
- Pouca fiscalização.



OPPORTUNITIES

OPORTUNIDADES

- Poucas ATTs e recicadoras
- Obrigatoriedade de destinação adequada.
- Ganho de mercado por menor preço.

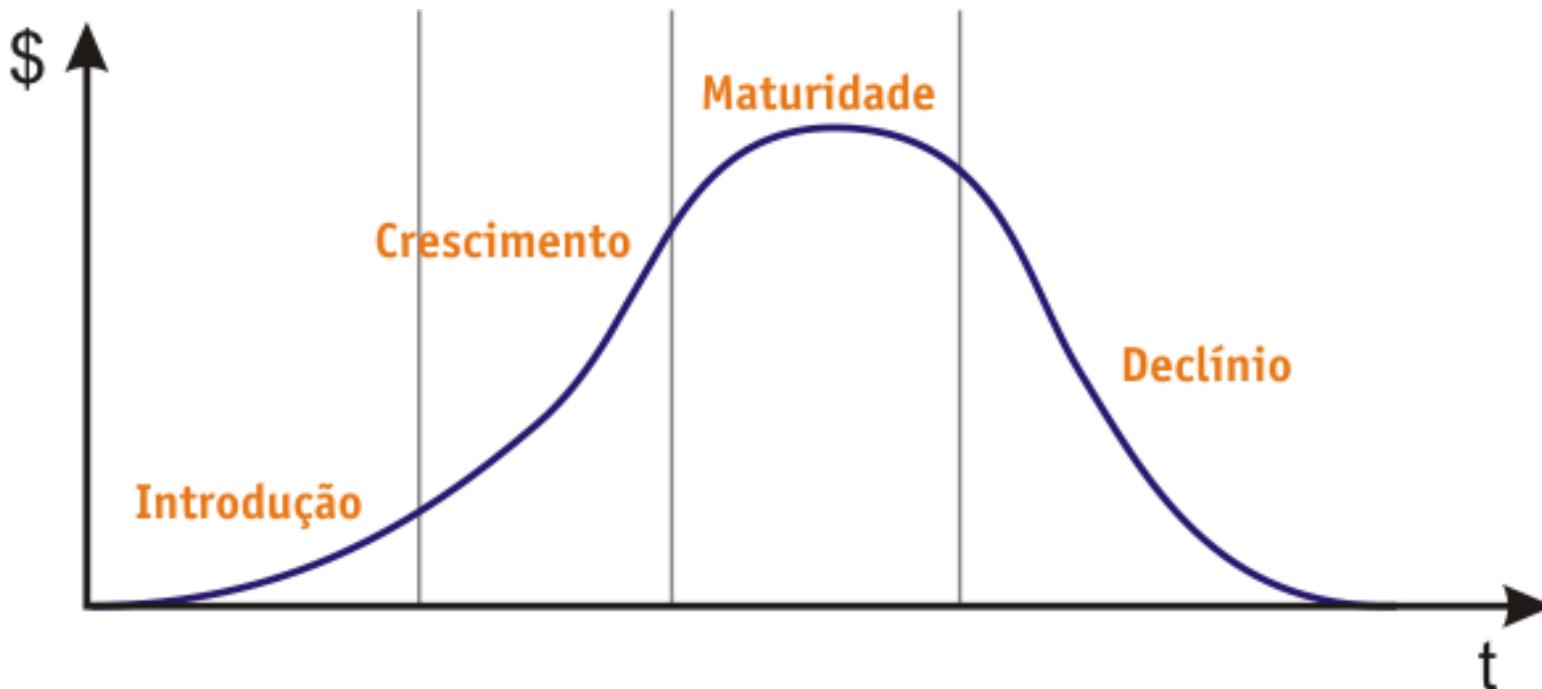


THREATS

AMEAÇAS

- Mudança cultural - Dificuldade de reconhecimento dos benefícios do produto.

CICLO DE VIDA DO PRODUTO



Criado por Paulo Henrique Ladeira

MATRIZ BCG

- Trata-se de uma análise gráfica desenvolvida pela Boston Consulting Group em 1970.
- Seu objetivo é suportar a análise de portfólio de produtos ou de unidades de negócio baseado no conceito de ciclo de vida do produto.
- Tem como objetivo auxiliar na definição de alocação de recursos em atividades como gestão de marca e marketing.

MATRIZ BCG



CLASSIFICAÇÃO DOS QUADRANTES

- **Em questionamento** ("ponto de interrogação" ou "criança-problemática")

- Ruim fluxo de caixa => altos investimentos e apresenta baixo retorno sobre ativos e tem baixa participação de mercado.
- Se nada é feito para mudar a participação de mercado, pode absorver um grande investimento e depois se tornar um "abacaxi".
- Por outro lado, por estar em um mercado de alto crescimento pode-se tornar um produto "estrela".



CLASSIFICAÇÃO DOS QUADRANTES

○ Estrela:

- Exige grandes investimentos e são referências no mercado, gerando receitas (ainda que não muito elevadas) e com taxas de crescimento potencialmente elevadas.
- Ficam frequentemente em equilíbrio quanto ao fluxo de caixa.
- Entretanto, a participação de mercado deve ser mantida, pois pode-se tornar numa "vaca leiteira" se não houver perda de mercado.



CLASSIFICAÇÃO DOS QUADRANTES

○ Vaca leiteira:

- Os lucros e a geração de caixa são altos.
- Como o crescimento do mercado é baixo, não são necessários grandes investimentos.
- Pode ser a base de uma empresa, já que a empresa detém uma quota de mercado considerável.



CLASSIFICAÇÃO DOS QUADRANTES



○ Abacaxi :

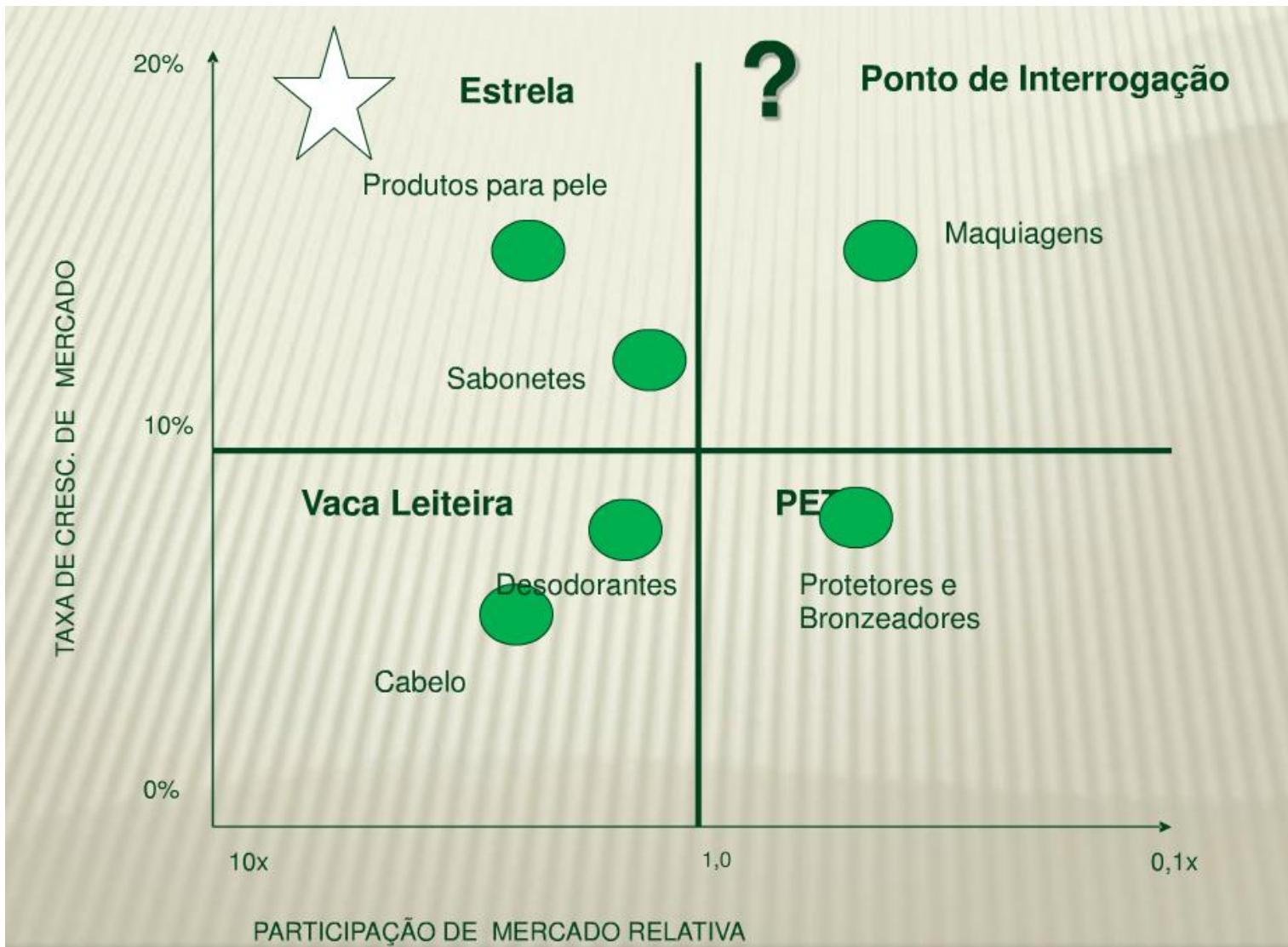
- Os "abacaxis" devem ser evitados e minimizados numa empresa.
- Cuidado com os caros planos de recuperação. Invista se for possível na recuperação, senão desista do produto.
- A baixa quota de mercado gera poucos lucros, mas estes estão associados a um baixo investimento devido ao crescimento do mercado praticamente nulo.
- A avaliação destes produtos deve ser feita de maneira a conseguir posicioná-los de maneira mais atrativa e rentável para a empresa, ou mesmo abandoná-los, quando a rentabilidade não seja de todo possível.

MATRIZ BCG

○ Desvantagens

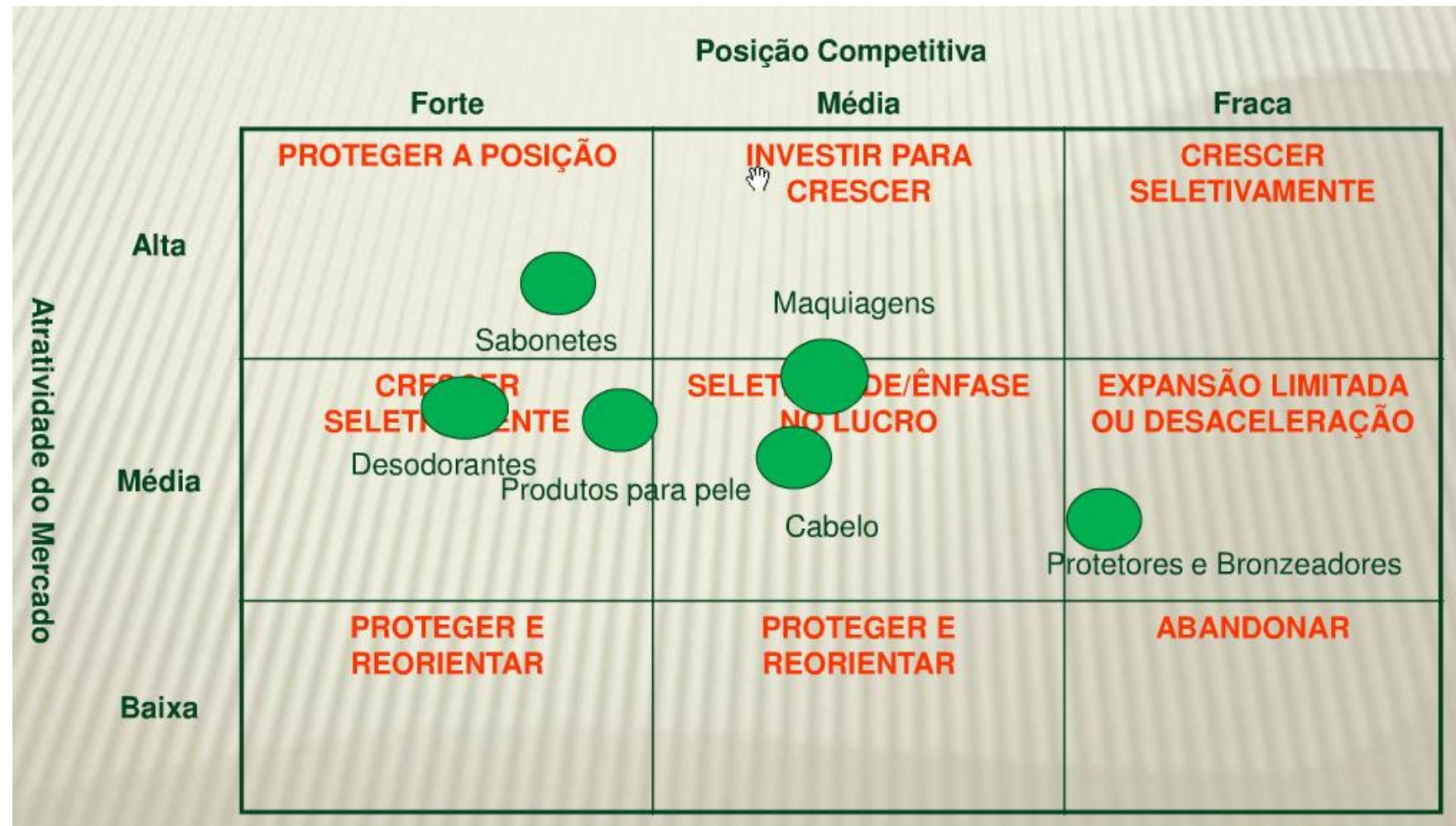
- Existe uma forte chance de ser tendencioso, se feito pelos “donos” do produto
- Alta participação de mercado não é o único fator de sucesso e
- Crescimento de mercado não é o único atrativo de mercado
- Nem sempre, um abacaxi é pior do que uma vaca leiteira

BCG: CASE BOTICÁRIO



Criado por Paulo Henrique Ladeira

BCG: CASE BOTICÁRIO



CASE BOTICÁRIO

- Fonte: <http://www.docstoc.com/docs/100640685/O-Boticrio>

PRODUTOS E A ORGANIZAÇÃO

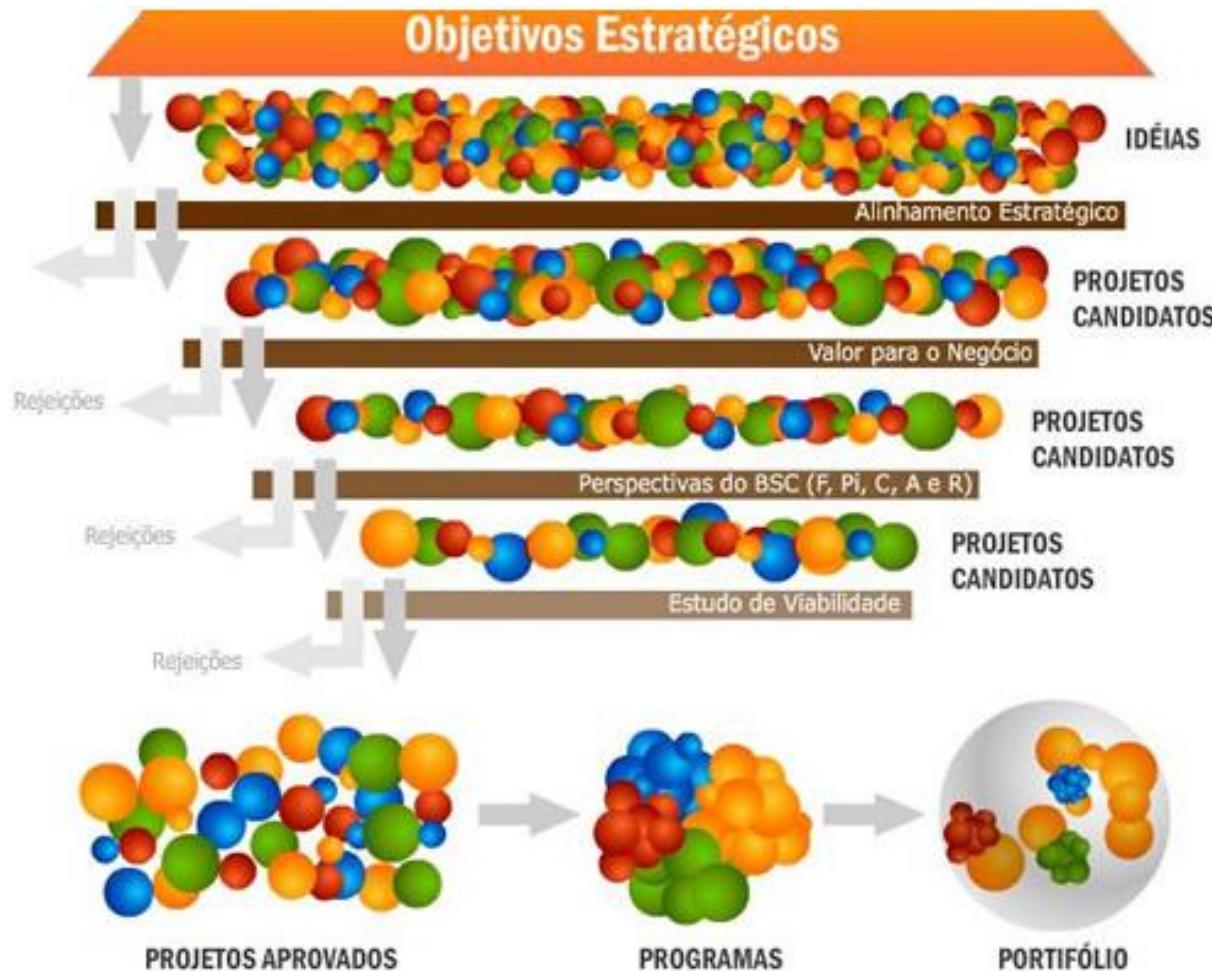
- De acordo com Bruce Henderson (criador da Matriz BCG):
 - "Para ter sucesso, uma empresa precisa ter um portfolio de produtos com diferentes taxas de crescimento e diferentes participações no mercado. A composição deste portfolio é uma função do equilíbrio entre fluxos de caixa. Produtos de alto crescimento exigem injecções de dinheiro para crescer. Produtos de baixo crescimento devem gerar excesso de caixa. Ambos são necessários simultaneamente."

PLANO DE NEGÓCIO

- Fonte:

[http://www.biblioteca.sebrae.com.br/bds/bds.nsf/797332C6209B4B1283257368006FF4BA/\\$File/NT000361B2.pdf](http://www.biblioteca.sebrae.com.br/bds/bds.nsf/797332C6209B4B1283257368006FF4BA/$File/NT000361B2.pdf)

O CICLO NA ORGANIZAÇÃO



- Fonte: <http://evertongomede.blogspot.com.br/2010/11/01/archive.html>

O TRABALHO NAS ORGANIZAÇÕES

- Dividido em duas categorias:
 - Operações rotineiras (guiadas por processo);
 - Projetos.
- Ambas são realizadas por pessoas, contém restrição de recursos e deve ser planejado, executado e controlado.

PROJETOS X OPERAÇÕES

	Operações	Projetos
Duração	Contínuas	Temporários
Execução	Repetitivas	Únicos
Objetivo	Manter e melhorar	Entregar e terminar
Forma de execução	Por processos	Projetos

GESTÃO DE PROJETOS – CENÁRIO ATUAL



Abrangência Geográfica



Time to market



Requisitos de qualidade



Complexidade técnica



Investimento



Disponibilidade de recursos

GESTÃO DE PROJETOS – CENÁRIO ATUAL

Projeto	País	Descrição	BI US\$
High Speed 2	Inglaterra	Criar uma ferrovia de alta velocidade ligando Londres e o West Midlands, e ligando Londres ao Norte da Inglaterra e Escocia.	45.6
Masdar City	Emirados Árabes	O projeto de Abu Dhabi é criar uma cidade com energia renovável com foco nas tecnologias limpas.	22
Rio - São Paulo - Campinas High Speed Rail	Brasil	Planejamento de construir uma ferróvia de alta velocidade conectando São Paulo e Rio de Janeiro.	20
Olimpíadas de 2016	Brasil	Olimpíadas no Brasil	14

Criado por Paulo Henrique Ladeira

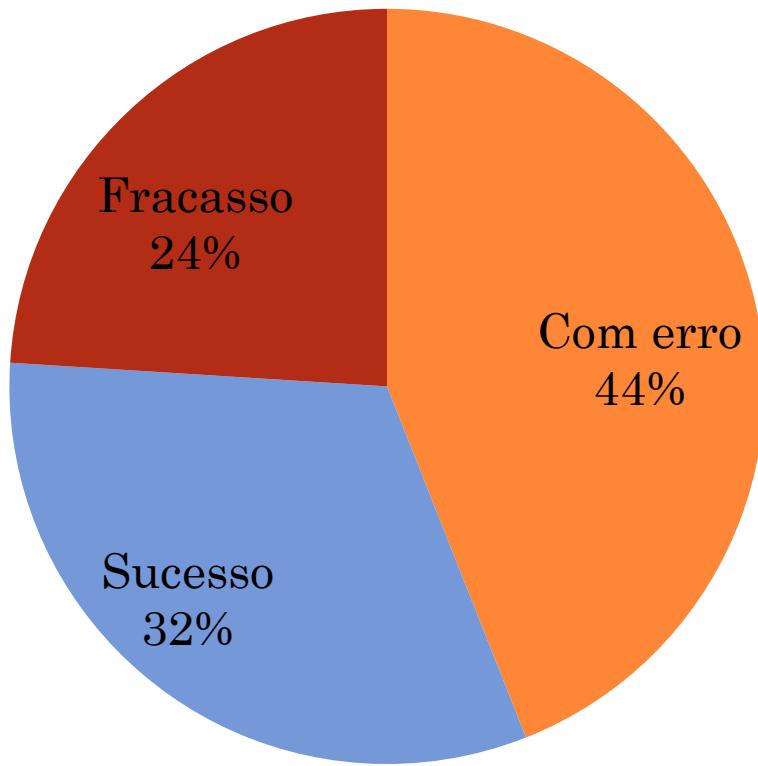
Fonte: Top 100 Global Infrastructure Projects, CG/LA Infrastructure, November 2010

GESTÃO DE PROJETOS – CENÁRIO ATUAL

- Estimativas indicam que **35%** do PIB mundial é gasto com projetos (480 Bilhões).
- Aproximadamente **16,5 milhões** de profissionais trabalhando diretamente com GP.

GESTÃO DE PROJETOS

CHAOS Report



Criado por Paulo Henrique Ladeira

PROGRAMAS

- Conjunto de projetos relacionados, gerenciados de forma coordenada para se obter benefícios não atingíveis se administrados isoladamente.
- Um projeto pode ou não fazer parte de um programa. Porém, um programa sempre será formado por diversos projetos.
- Os projetos compostos em um programa devem ter relação com o resultado comum ou capacidade coletivo. Se a relação é somente entre tecnologia, cliente, setor, deve ser gerido como portfólio.

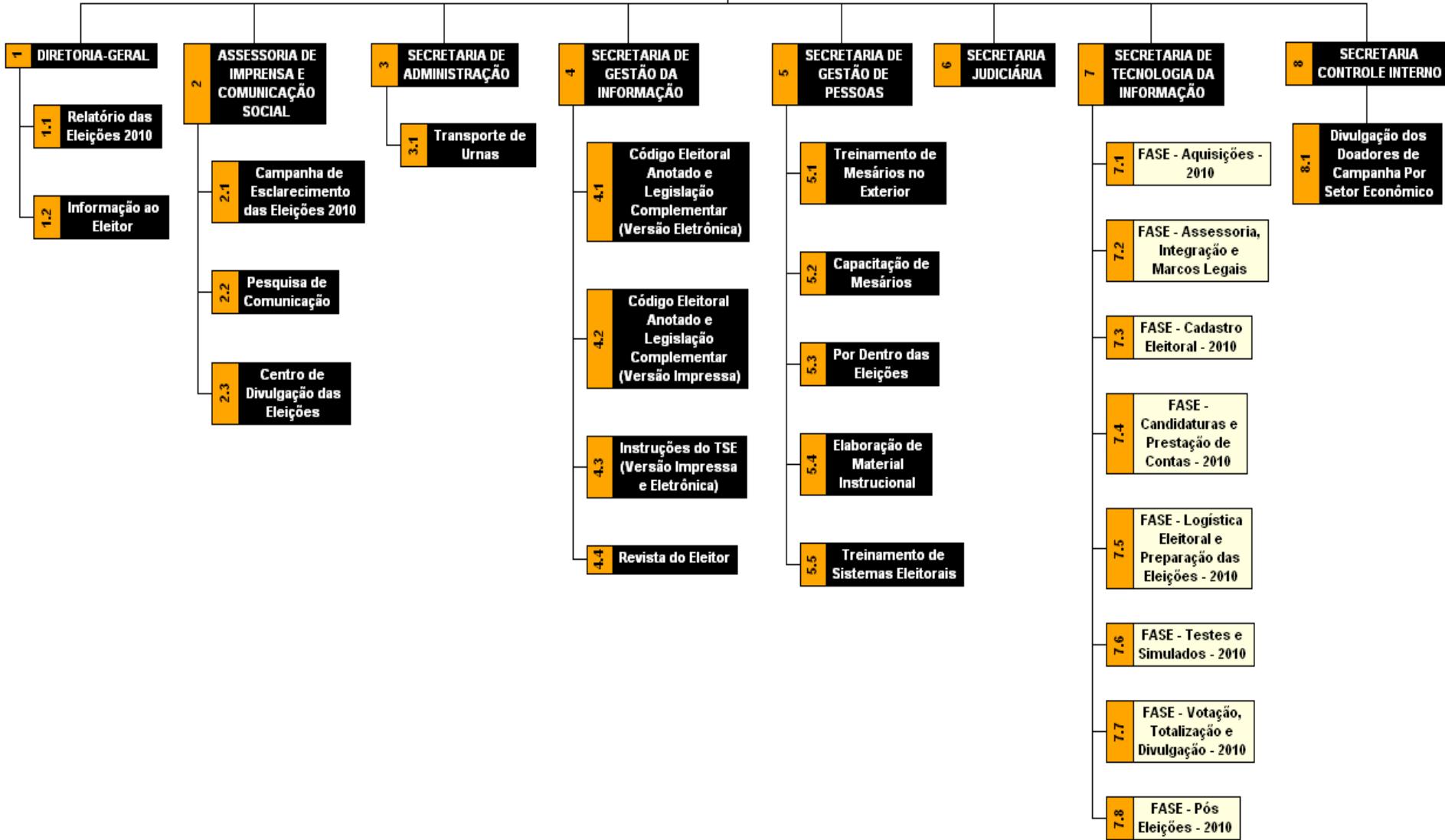
GERENCIAMENTO DE PROGRAMAS

- Podem ser estruturados de duas maneiras:
 - **Pela fragmentação de uma ação muito abrangente** em diversos projetos, gerenciados um a um de modo que, quando todos forem finalizados, realizem um plano geral.
 - **Pelo agrupamento de muitos projetos executados em paralelo** que acabam revelando alguns objetivos comuns, de modo a criar resultados coordenados e convergentes.

GERENCIAMENTO DE PROGRAMAS

- Os programas não incluem aspectos operacionais e a descrição detalhada de atividades.
- Os programas dependem dos projetos a eles subordinados.
- Um programa é formalmente finalizado apenas quando todos os projetos que incluem são completados.

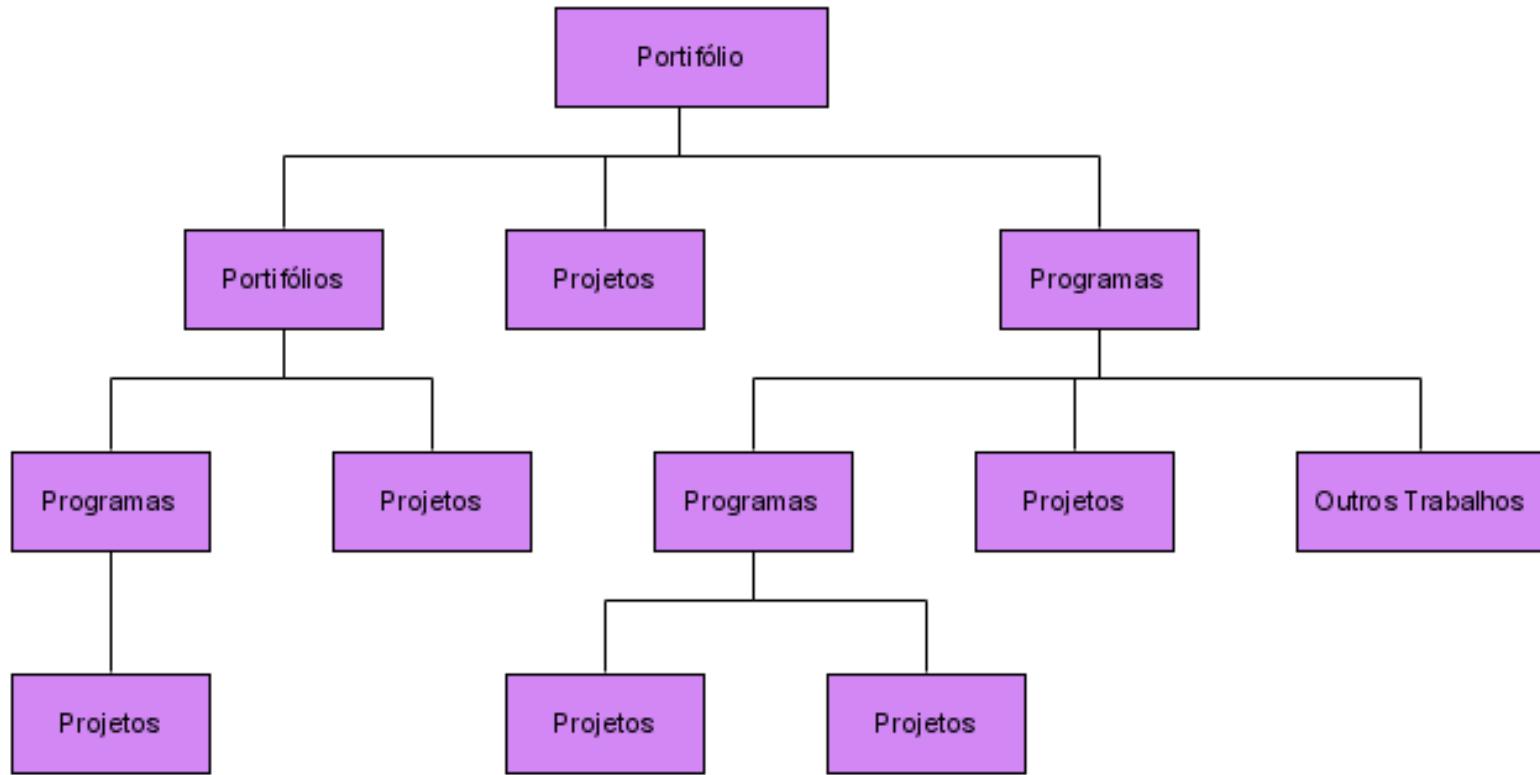
0 Programa Eleições
2010.mpp



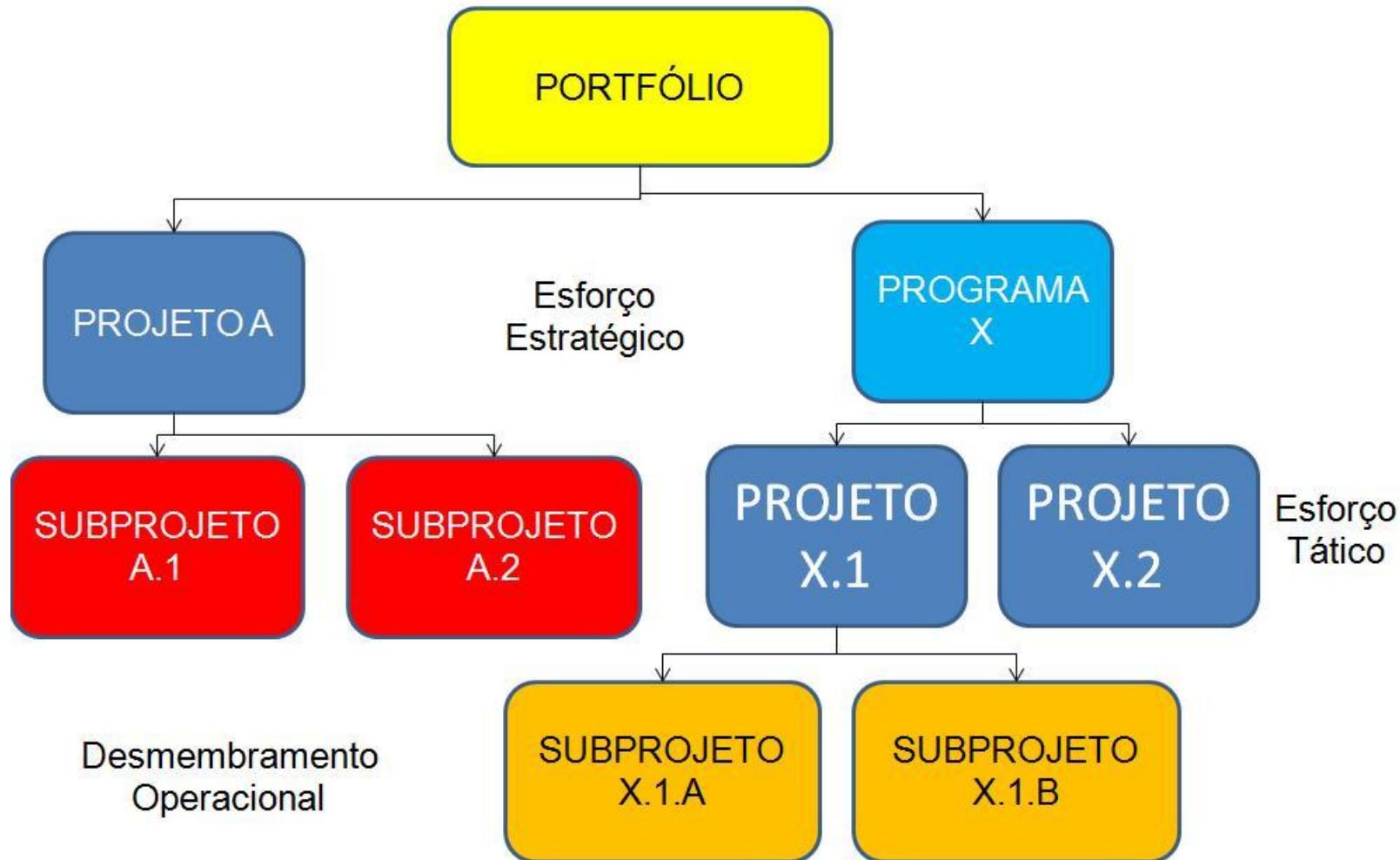
PORTFÓLIO

- Um portfólio é uma coleção de programas e/ ou projetos agrupados de modo a facilitar a sua integração em torno de objetivos comuns.
- Não existe a necessidade dos projetos/programas existentes terem relacionamento/dependência.
- Como ocorre com os programas, a parte operacional não é realizada no nível do portfólio.

PORTFÓLIO



PORTFÓLIO



DEFINIÇÃO DE PROJETOS

- “É um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo.”

Fonte: PMBOK

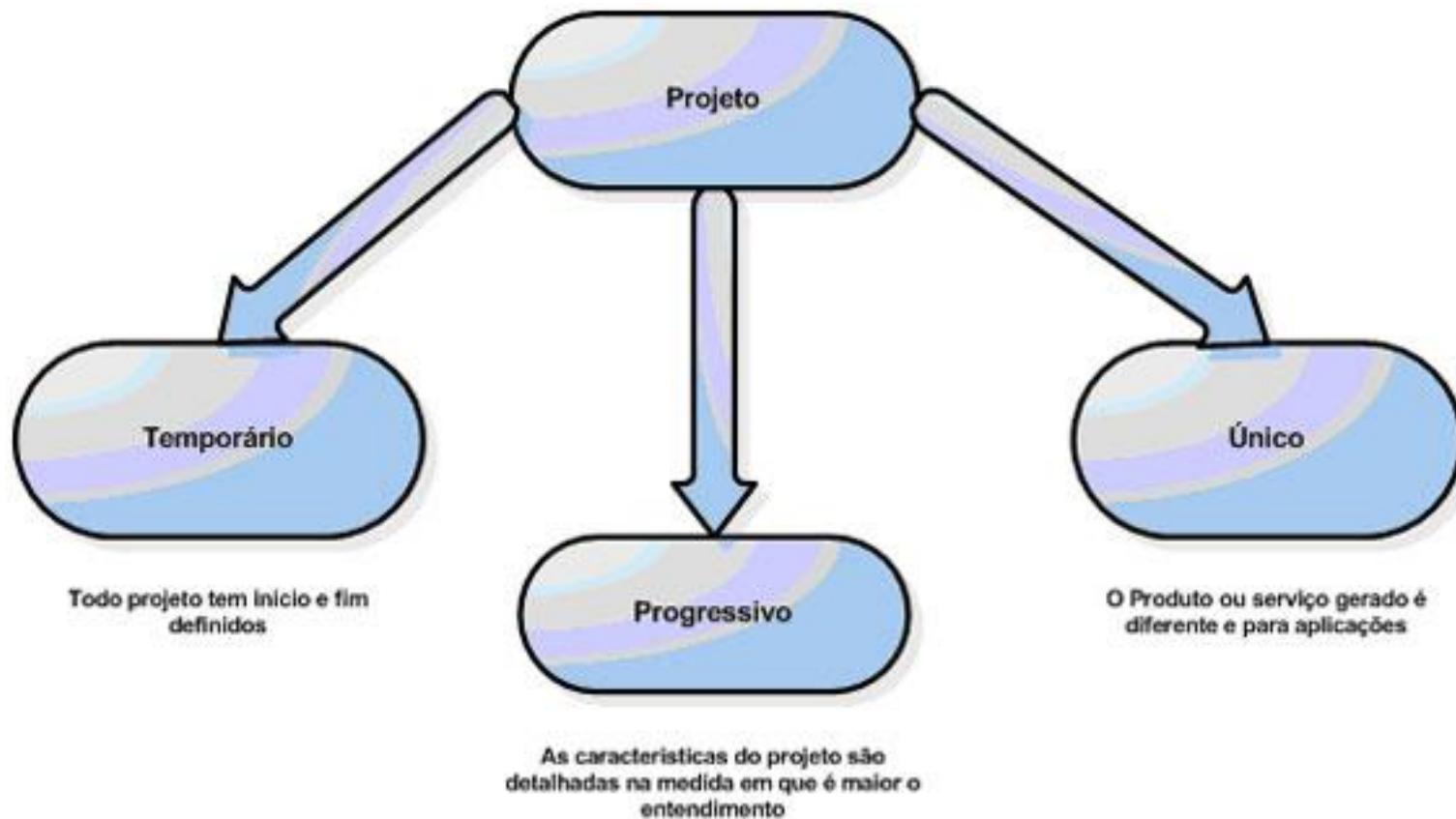
- “Um projeto consiste num esforço temporário empreendido com um objetivo pré-estabelecido, definido e claro, seja criar um novo produto, serviço, processo. Tem início, meio e fim definidos, duração e recursos limitados, numa seqüência de atividades relacionadas.”

Fonte: Avellareduarte

GESTÃO DE PROJETOS

- Os projetos são normalmente autorizados como resultado de uma ou mais considerações estratégicas. Podendo ser:
 - uma demanda de mercado;
 - necessidade organizacional;
 - solicitação de um cliente;
 - avanço tecnológico ou
 - requisito legal.
- As principais características dos projetos são:
 - temporários, possuem um início e um fim definidos.
 - planejados, executado e controlado.
 - entregam produtos, serviços ou resultados exclusivos.
 - desenvolvidos em etapas e continuam por incremento com uma elaboração progressiva.
 - realizados por pessoas.
 - com recursos limitados.

GESTÃO DE PROJETOS

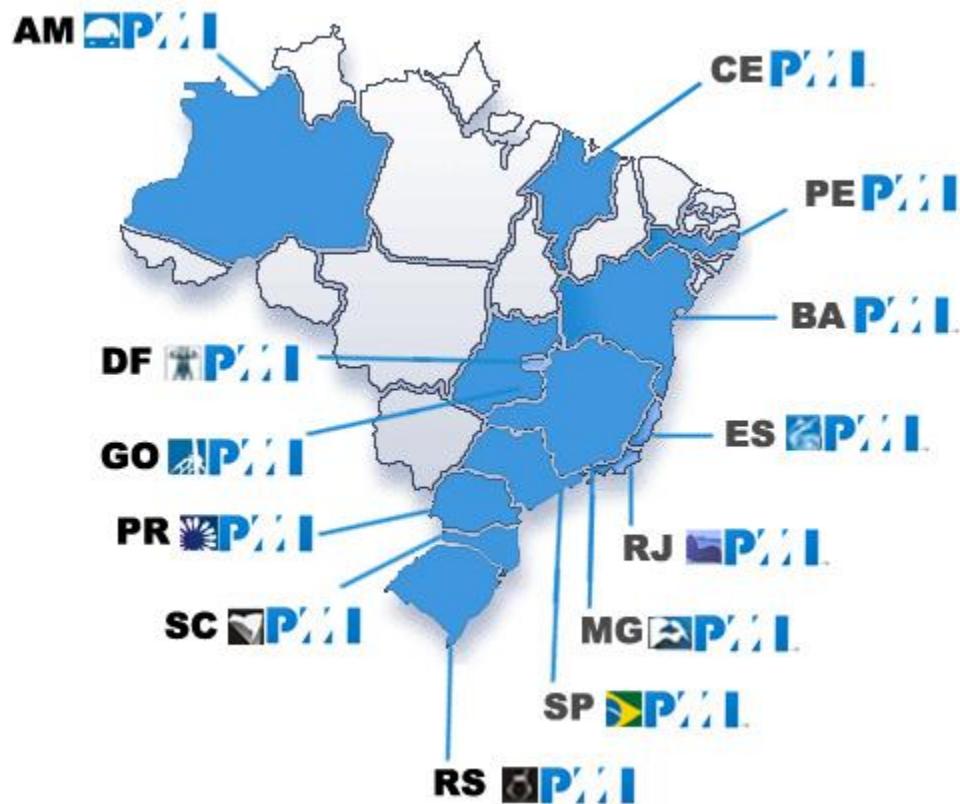


PROJETOS X PROGRAMAS X PORTFÓLIOS

	Projeto	Programa	Portfólio
Objetivo	Producir entregáveis	Implementar mudança estratégica	Coordenar, otimizar e alinhar
Benefício para a organização	Escopo, Prazo, Qualidade e Custo	Integração de diferentes projetos	Implementar a estratégia de forma otimizada

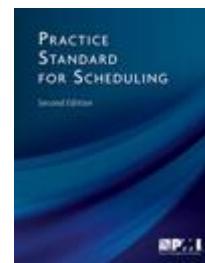
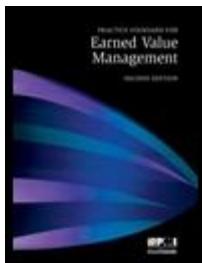
- Trata-se de uma organização sem fins lucrativos que promove a profissão de gerenciamento de projetos por meio de padrões e certificações mundialmente reconhecidos.
 - Mais de meio milhão de membros e titulares de credenciais, em mais de 180 países.
-
- <http://www.pmi.org/>
 - <http://brasil.pmi.org/>
 - <http://www.pmimg.org.br/>

GESTÃO DE PROJETOS - PMI



Criado por Paulo Henrique Ladeira

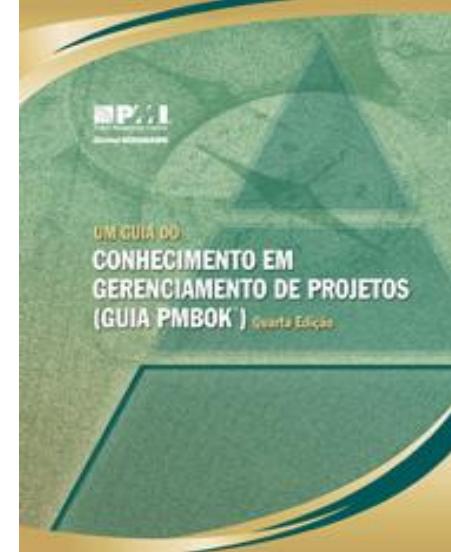
GESTÃO DE PROJETOS - PMI



Criado por Paulo Henrique Ladeira

GESTÃO DE PROJETOS - PMI

- PMBOK – Project Management Body of Knowledge ou corpo de conhecimento em gerenciamento de projetos
- Padronização dos termos e das melhores práticas de GP
- Desenvolvido pela contribuição de profissionais e estudantes de GP do mundo
- Desenvolvido para qualquer setor
- Traduzido para 10 idiomas



ado por Paulo Henrique Ladeira

GESTÃO DE PROJETOS - PMI

The PMI Family of Credentials

	CAPM®	PMI-SP®	PMI-RMP®	PMP®	PgMP®
Full Name	Certified Associate in Project Management	PMI Scheduling Professional	PMI Risk Management Professional	Project Management Professional	Program Management Professional
Project Role	Contributes to project team	Develops and maintains project schedule	Assesses and identifies risks and mitigates threats and capitalizes opportunities	Leads and directs project teams	Achieves an organizational objective through defining and overseeing projects and resources
Eligibility Requirements	High school diploma/global equivalent AND 1,500 hours experience OR 23 hours pm education	High school diploma/global equivalent 5,000 hours project scheduling experience 40 hours project scheduling education OR Bachelor's degree/global equivalent 3,500 hours project scheduling experience 30 hours project scheduling education	High school diploma/global equivalent 4,500 hours project risk management experience 40 hours project risk management education OR Bachelor's degree/global equivalent 3,000 hours project risk management experience 30 hours project risk management education	High school diploma/global equivalent 5 years project management experience 35 hours project management education OR Bachelor's degree/global equivalent 3 years project management experience 35 hours project management education	High school diploma/global equivalent 4 years project management experience 7 years program management experience OR Bachelor's degree/global equivalent 4 years project management experience 4 years program management experience
Steps to Obtaining Credential	application process + multiple-choice exam	application process + multiple-choice exam	application process + multiple-choice exam	application process + multiple-choice exam	3 evaluations – application panel review + multiple-choice exam + multi-rater assessment
Exam Information	3 hours; 150 questions	3.5 hours; 170 questions	3.5 hours; 170 questions	4 hours; 200 questions	4 hours; 170 questions
Fees	US\$225 PMI member (US\$300 non-member)	US\$520 PMI member (US\$670 non-member)	US\$520 PMI member (US\$670 non-member)	US\$405 PMI member (US\$555 non-member)	US\$1,500 PMI member (US\$1,800 non-member)
Credential Maintenance Cycles and Requirements	5 years; re-exam	3 years; 30 PDUs in project scheduling	3 years; 30 PDUs in risk management	3 years; 60 PDUs	3 years; 60 PDUs

For complete details on eligibility requirements, read the [online handbook](#) for the credential for which you plan to apply.

GESTÃO DE PROJETOS - PMI

Distribuição Regional em 1997



Criado por Paulo Henrique Ladeira

GESTÃO DE PROJETOS - PMI

Distribuição Regional em 2009

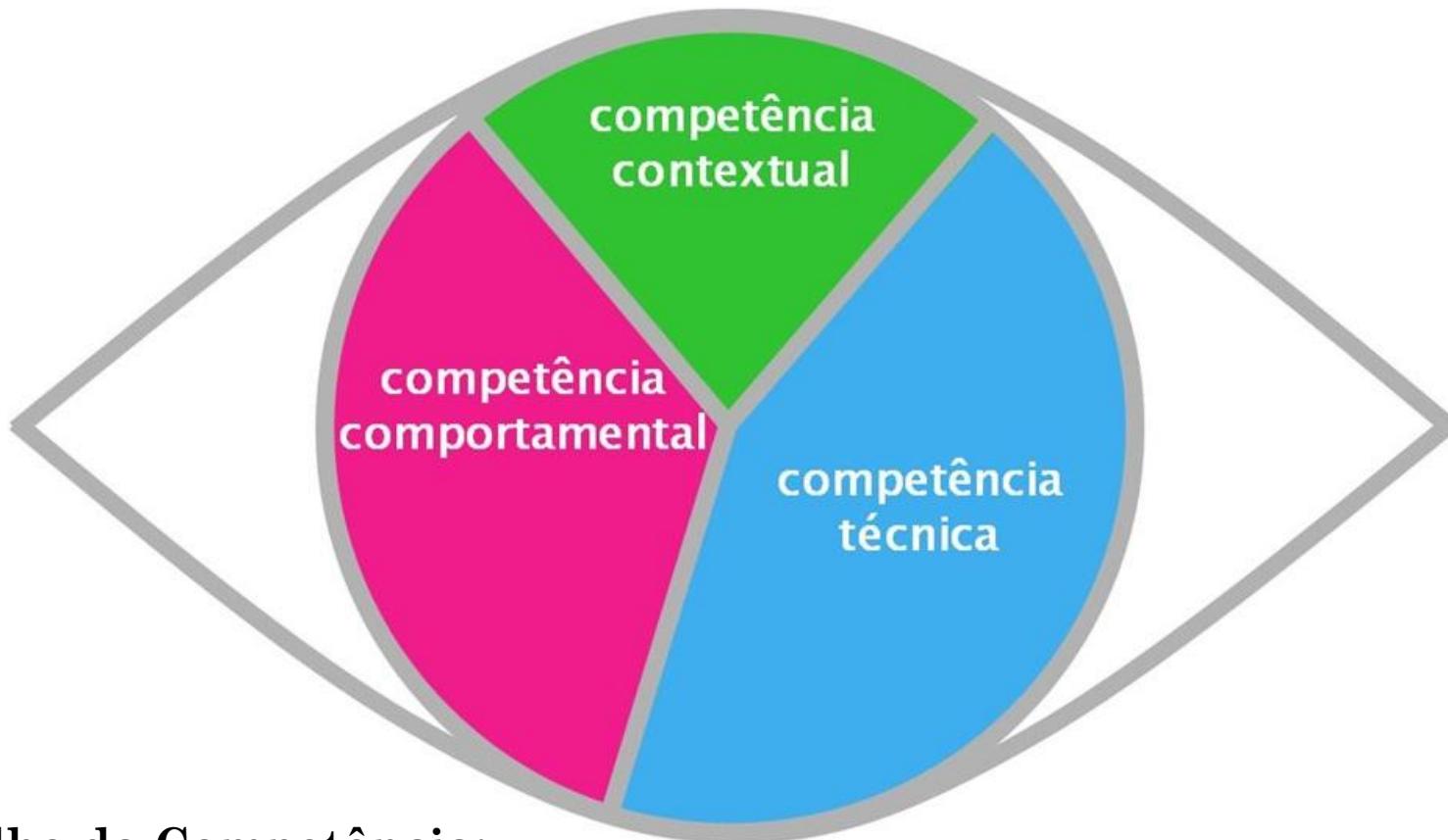


Criado por Paulo Henrique Ladeira

GESTÃO DE PROJETOS - IPMA

- O IPMA (International Project Management Association) é o único modelo de certificação baseado em competências.

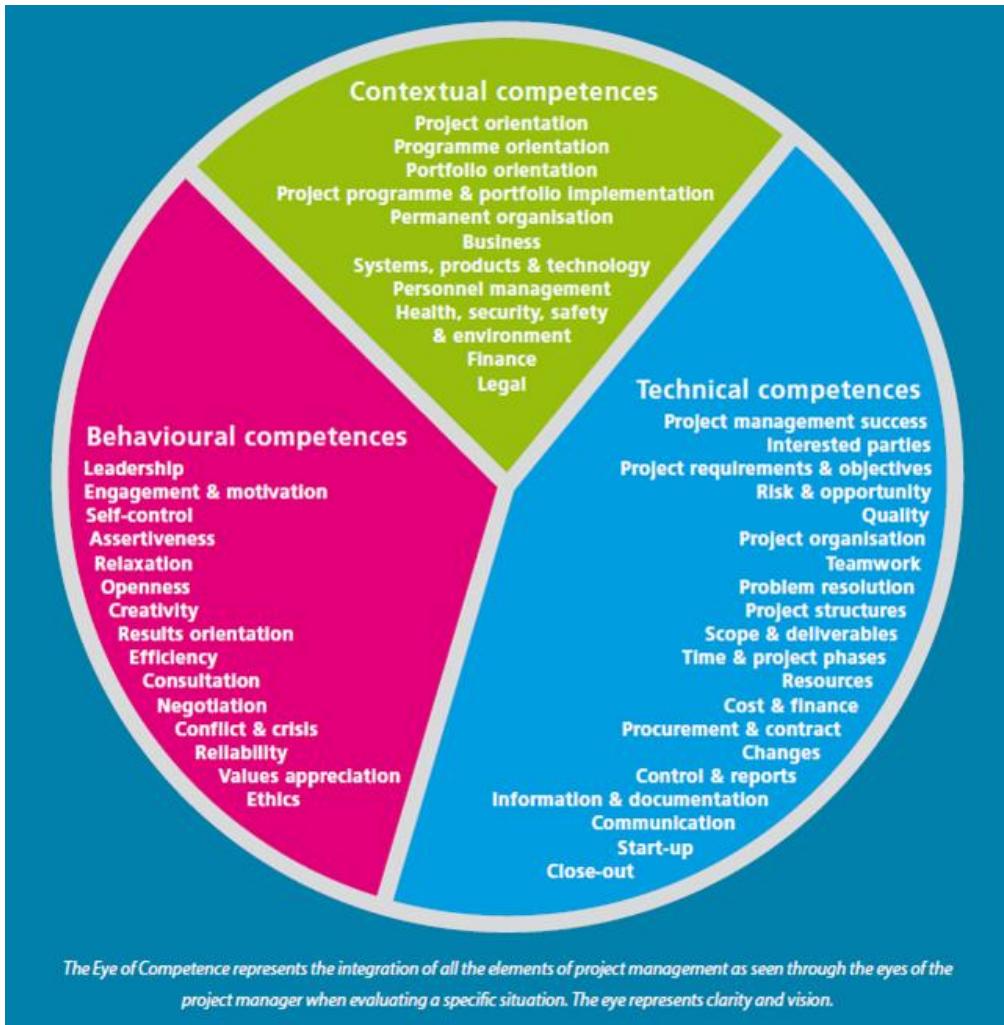
GESTÃO DE PROJETOS - IPMA



Olho da Competência:

representa o olhar IPMA sobre as melhores práticas e competências dos Gerentes de Projetos. Simboliza o conjunto representado pelos 46 Elementos do ICBv3, em que 20 Elementos são da Competência Técnica, 15 da Competência Comportamental e 11 da Competência Contextual.

GESTÃO DE PROJETOS - IPMA





GESTÃO DE PROJETOS

○ Certificação

Title	Capabilites	Certification Process			Validity
		Stage 1	Stage 2	Stage 3	
Certified Projects Director (IPMA Level A)		A	Program / Portfolio description	Program / Portfolio report	
Certified Senior Project Manager (IPMA Level B)	Competence = applied knowledge + relevant experience + professional behaviour	B	Project description	Project report	5 years
Certified Project Manager (IPMA Level C)		C	Exam	Short project report	
Certified Project Management Associate (IPMA Level D)	Know-ledge	D	Application, curriculum vitae, self assesment	Exam	5 years

Criado por Paulo Henrique Ladeira

GESTÃO DE PROJETOS

- O gerente de projetos



GESTÃO DE PROJETOS

O falso gerente de projetos

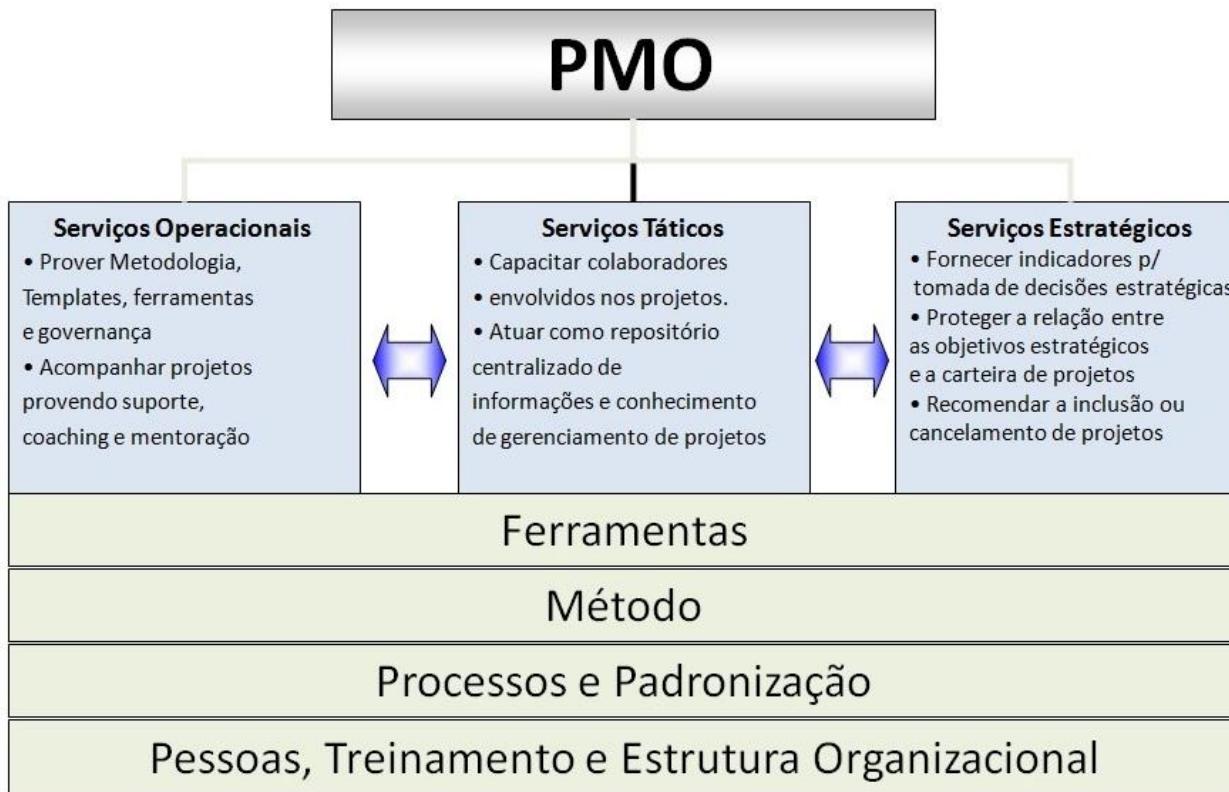


ESCRITÓRIO DE GESTÃO DE PROJETOS (PMO)

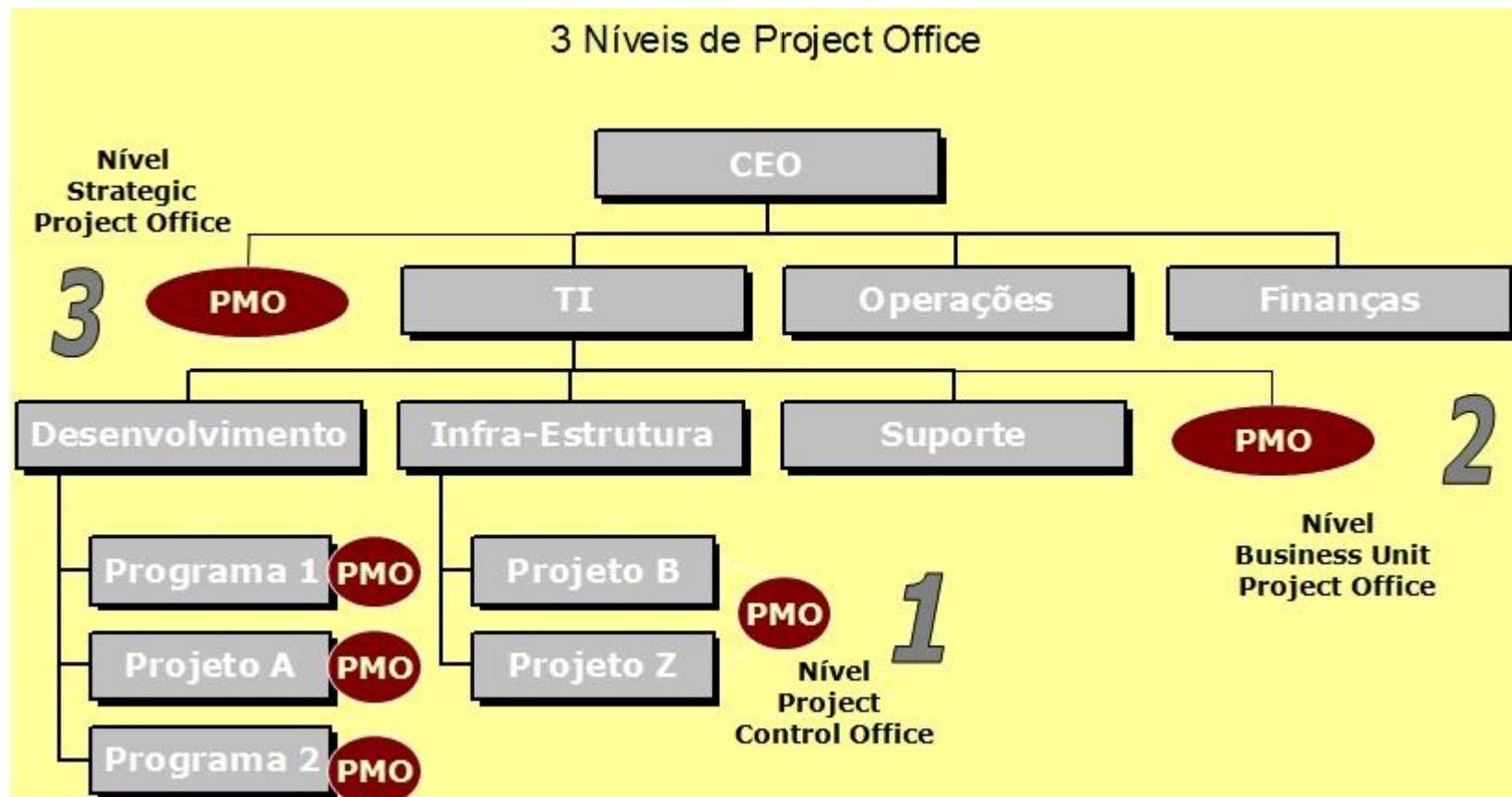
- Responsabilidades:

- Suporte ao gerente de projetos;
- Padronização;
- Auditorias e avaliações;
- Treinamentos diversos;
- Manter diretoria informada;
- Selecionar projetos.

ESCRITÓRIO DE GESTÃO DE PROJETOS (PMO)

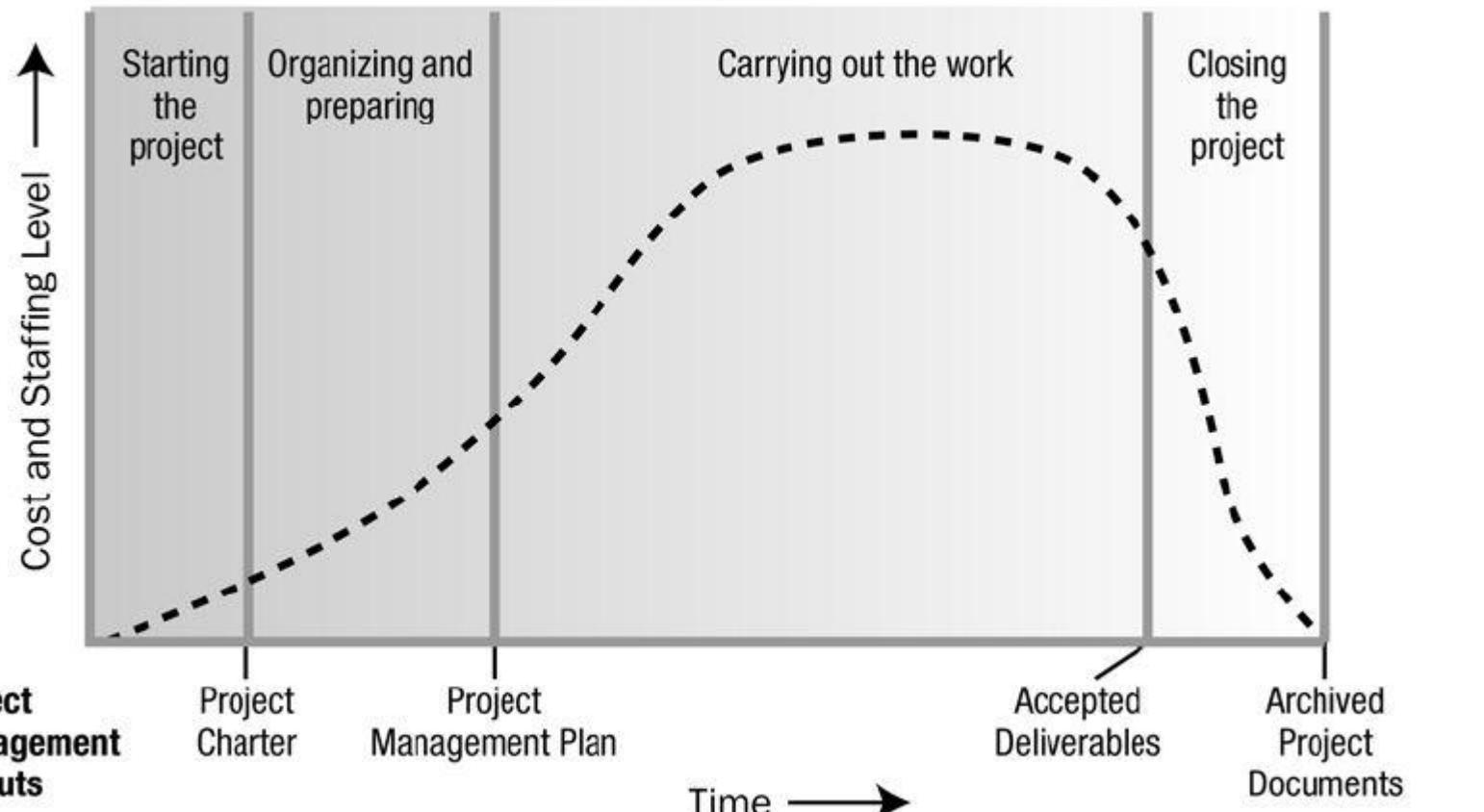


ESCRITÓRIO DE GESTÃO DE PROJETOS (PMO)



Criado por Paulo Henrique Ladeira

ESFORÇO TÍPICO EM PROJETOS



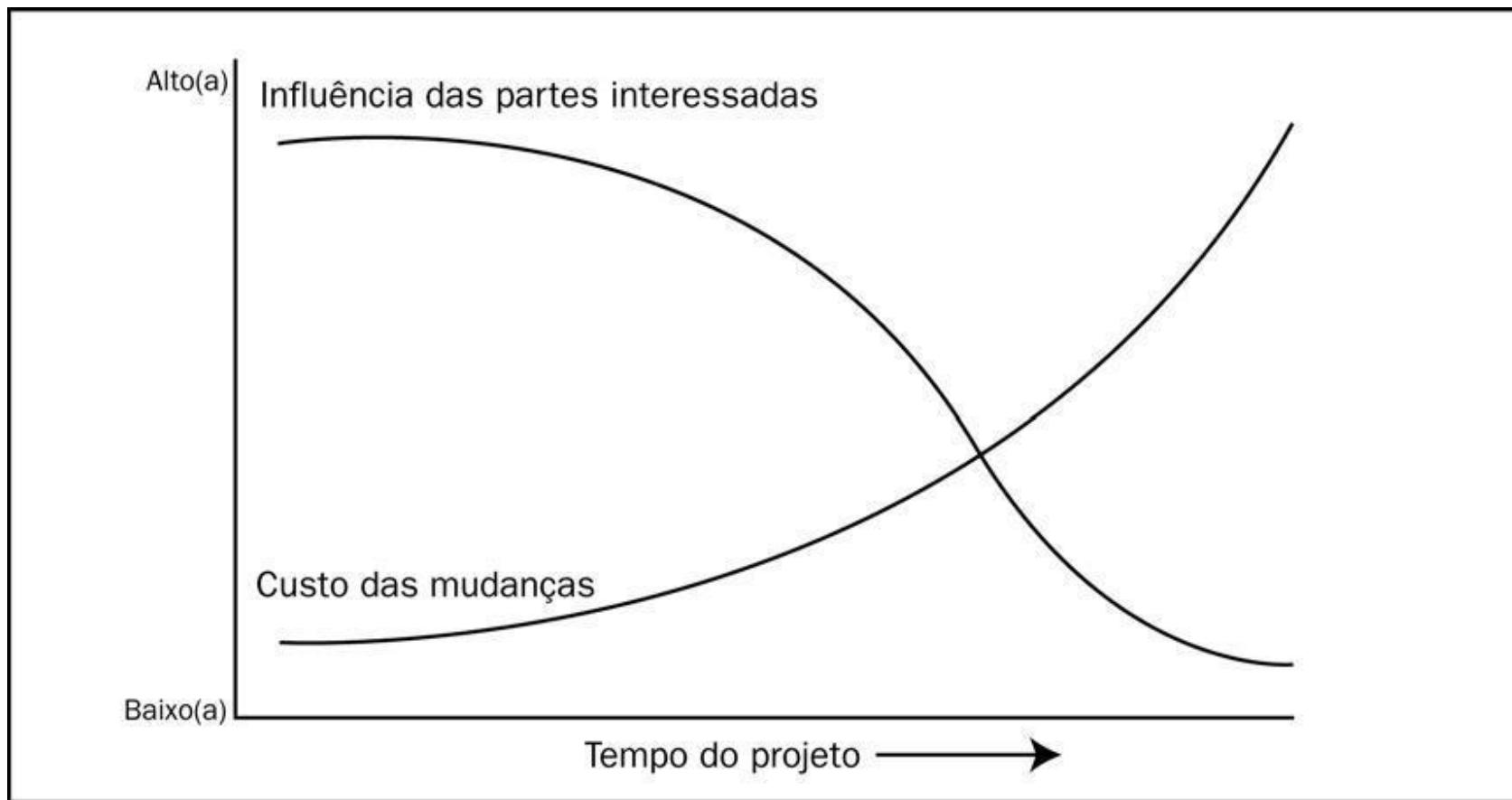
Criado por Paulo Henrique Ladeira

STAKEHOLDERS

- Stakeholders refere-se ás **partes interessadas do projeto**, que podem ser **pessoas, grupos ou mesmo outras empresas**
- Contém interesses que podem ser afetados diretamente de forma positiva ou negativa com a execução e conclusão do projeto.
- Exercem influência sobre o projeto e seus resultados.
- A equipe de gerenciamento deve identificar os stakeholders, determinar suas necessidades e gerenciar isto, em busca de um projeto bem-sucedido.

Fonte: <http://www.linhadecodigo.com.br/artigo/1167/conceitos-basicos-de-gerenciamento-de-projetos.aspx>

STAKEHOLDERS E INFLUÊNCIA PARA MUDANÇAS



Criado por Paulo Henrique Ladeira

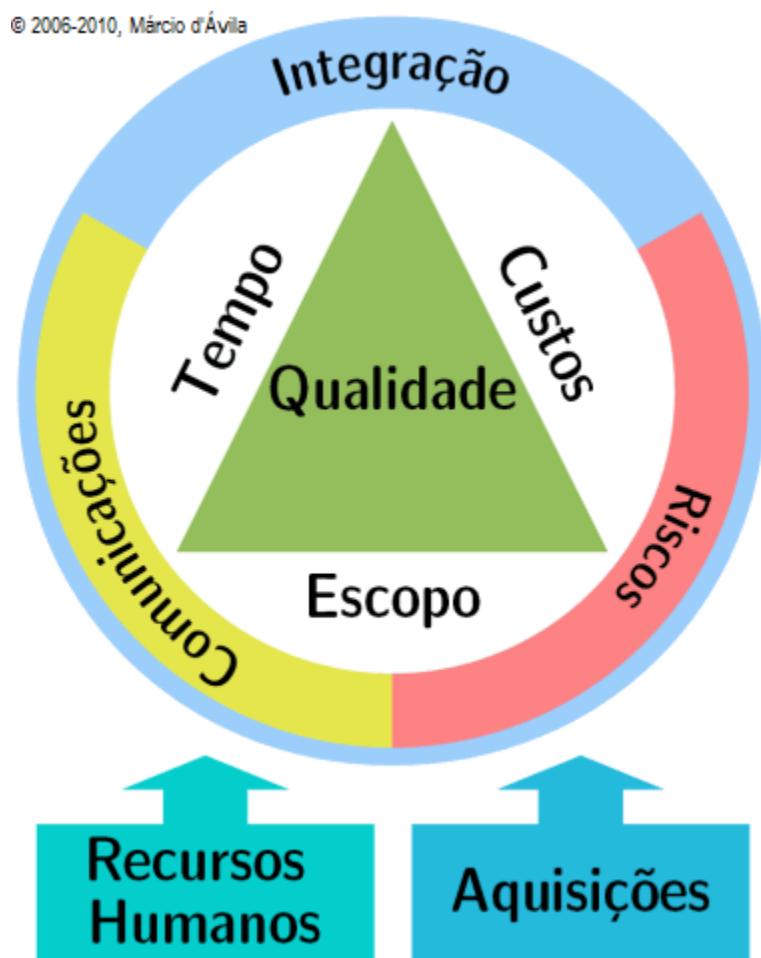
OS GRUPOS DE PROCESSOS

- O gerenciamento de projetos é a aplicação de **conhecimento, habilidades, ferramentas e técnicas** às atividades do projeto a fim de atender aos seus requisitos e metas.
- Agrupados em cinco grupos de processos:
 - Iniciação;
 - Planejamento;
 - Execução;
 - Controle;
 - Encerramento.

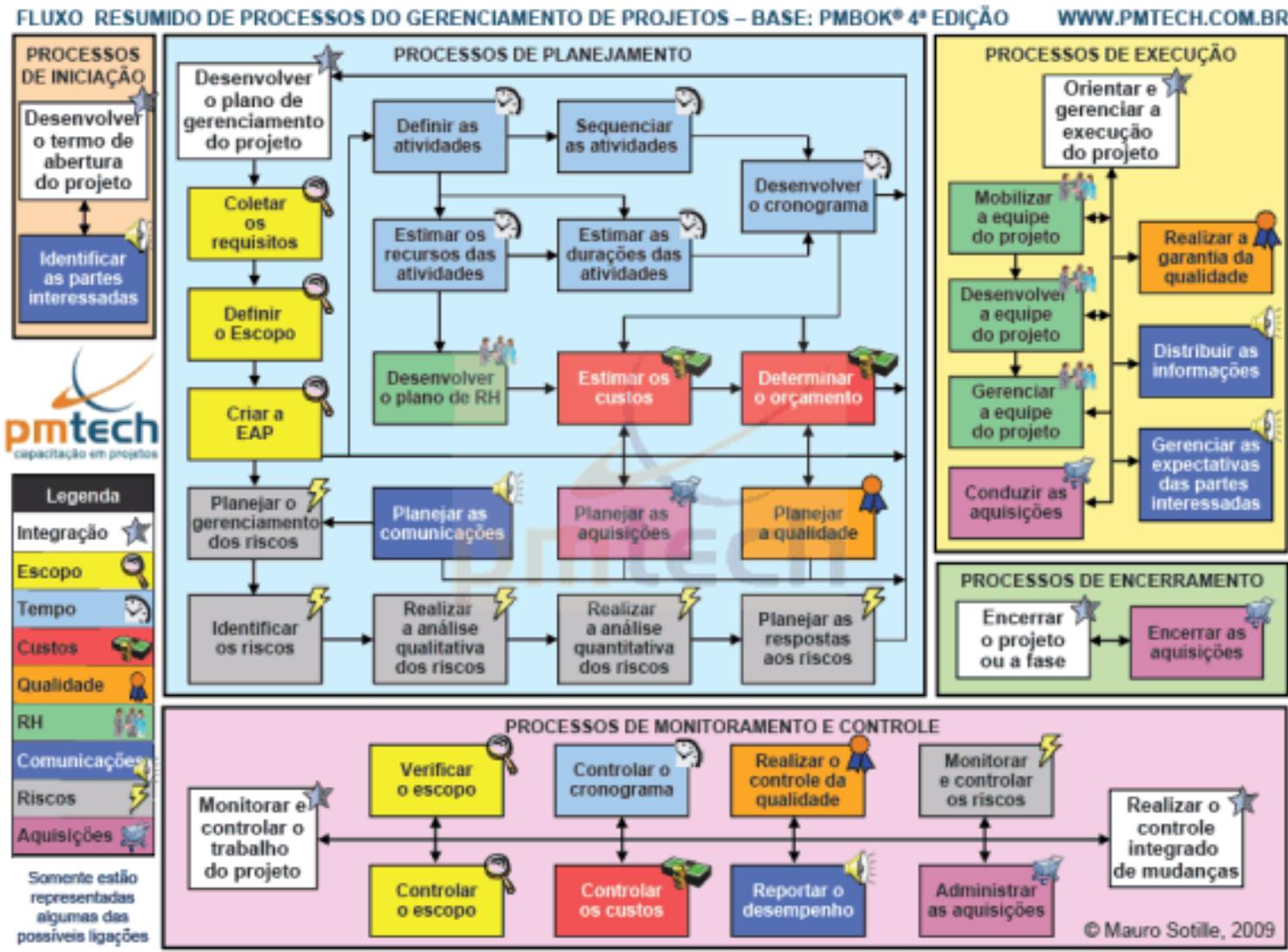


ÁREAS DE CONHECIMENTO DA GESTÃO DE PROJETOS

- A gestão de projetos é dividida em nove áreas de conhecimento:
 - Integração
 - Escopo
 - Tempo
 - Custos
 - Qualidade
 - Recursos humanos
 - Comunicações
 - Riscos
 - Aquisições



ÁREAS DE CONHECIMENTO X GRUPOS DE PROCESSOS



ÁREAS DE CONHECIMENTO X GRUPOS DE PROCESSOS



- Fonte: <http://www.mhavila.com.br/topicos/gestao/pmbok.html>

INICIAÇÃO DO PROJETO

- Fase preliminar da execução do projeto.
- Condições iniciais de um projeto bem sucedido são criadas.
- Fundamentos para execução do projeto são definidos.
- Caracterizada por:
 - Expectativas ainda pouco definidas.
 - Grande incerteza quanto à chance de sucesso.
 - Pressão por prazos.

INICIAÇÃO DO PROJETO

- Algumas das tarefas importantes que devem ser feitas nessa fase:
 - Definição do GP;
 - Definição da finalidade, objetivo e escopo do projeto;
 - Preparação e definição da organização do projeto;
 - Definição dos procedimentos de comunicação;
 - Planos iniciais são estabelecidos;
 - Definição do Project Charter ou Termo de abertura do projeto (TAP).

INICIAÇÃO DO PROJETO

- Algumas perguntas que precisam ser respondidas nessa fase:
 - Qual é o objetivo do projeto?
 - Qual o resultado/produto/serviço esperado?
 - Quem são os stakeholders chaves do projeto?
 - O que cada um espera do projeto?
 - Qual o prazo de execução?
 - Qual o orçamento aprovado para o projeto?
 - Onde o projeto será realizado?

INICIAÇÃO DO PROJETO

- Ver alguns modelos do termo de abertura:
 - <http://www.trainning.com.br/download/planodeprojeto.pdf>
 - http://www.pmtech.com.br/Escopo/Termo_de_abertura_Modelo.pdf

INICIAÇÃO DO PROJETO

○ Reunião de *kick off*

- Apresentar o objetivo do projeto;
- Formalizar o gerente do projeto a todos;
- Alinhar as entregas;
- Apresentar as metas em relação a custo, prazo e qualidade;
- Alinhar a estratégia da empresa com o projeto;
- Definir as interfaces do projeto;
- Definir atribuições da gerencia funcional e as equipes.

PLANEJAMENTO DO PROJETO



- Trata-se do processo de definição dos objetivos e planos do projeto.

- Nessa fase planejamos:
 - Escopo;
 - Requisitos de QA;
 - Prazos;
 - Orçamentos e fluxo de caixa;
 - Equipes e as suas atribuições;
 - Solicitações de mudanças;
 - Fluxos de comunicações;
 - Riscos;
 - Contratações.

PLANEJAMENTO DO PROJETO

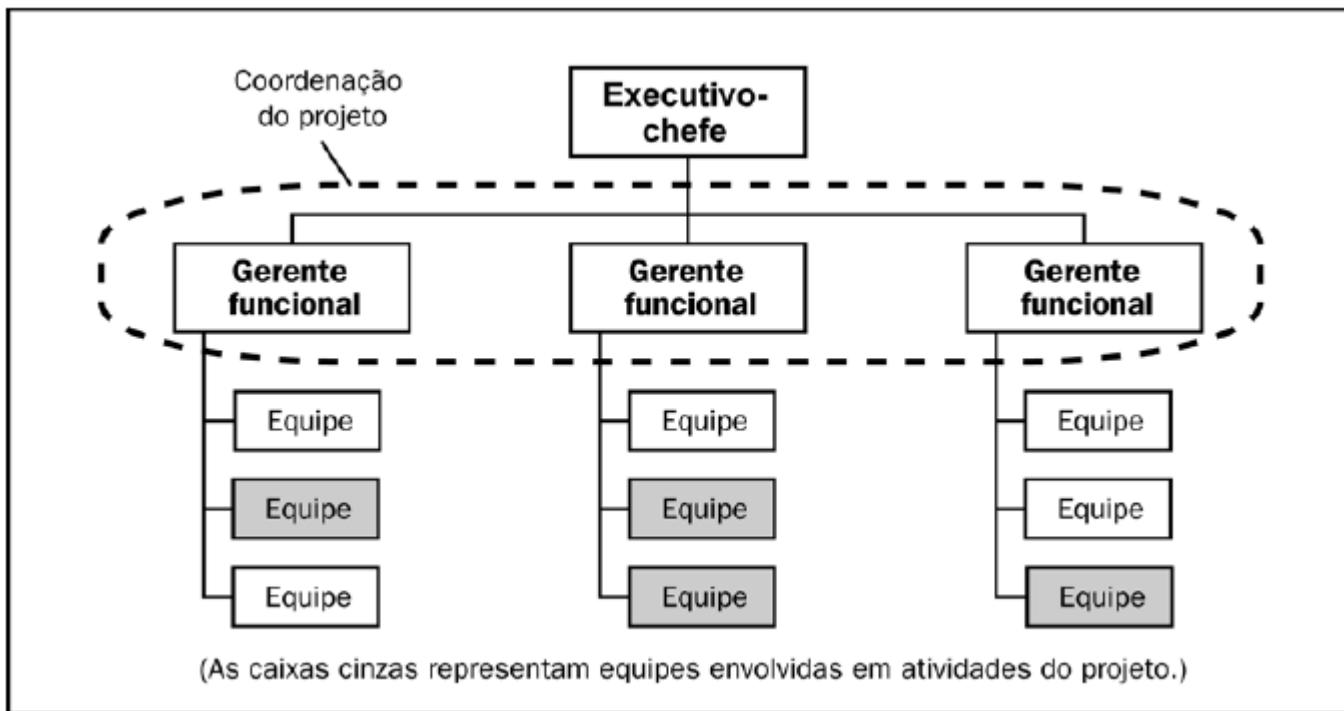
- Tem como objetivo elaborar o plano do projeto.
- Definir os marcos: linha de base para escopo, prazo, custo e qualidade do projeto.
- Aprovar plano do projeto junto aos stakeholders.

PLANEJAMENTO DO PROJETO - EQUIPE

- Para definir a equipe, após a aprovação do projeto, dependemos da estrutura da organização.
Que pode ser:
 - Funcional
 - Matricial
 - Por projetos

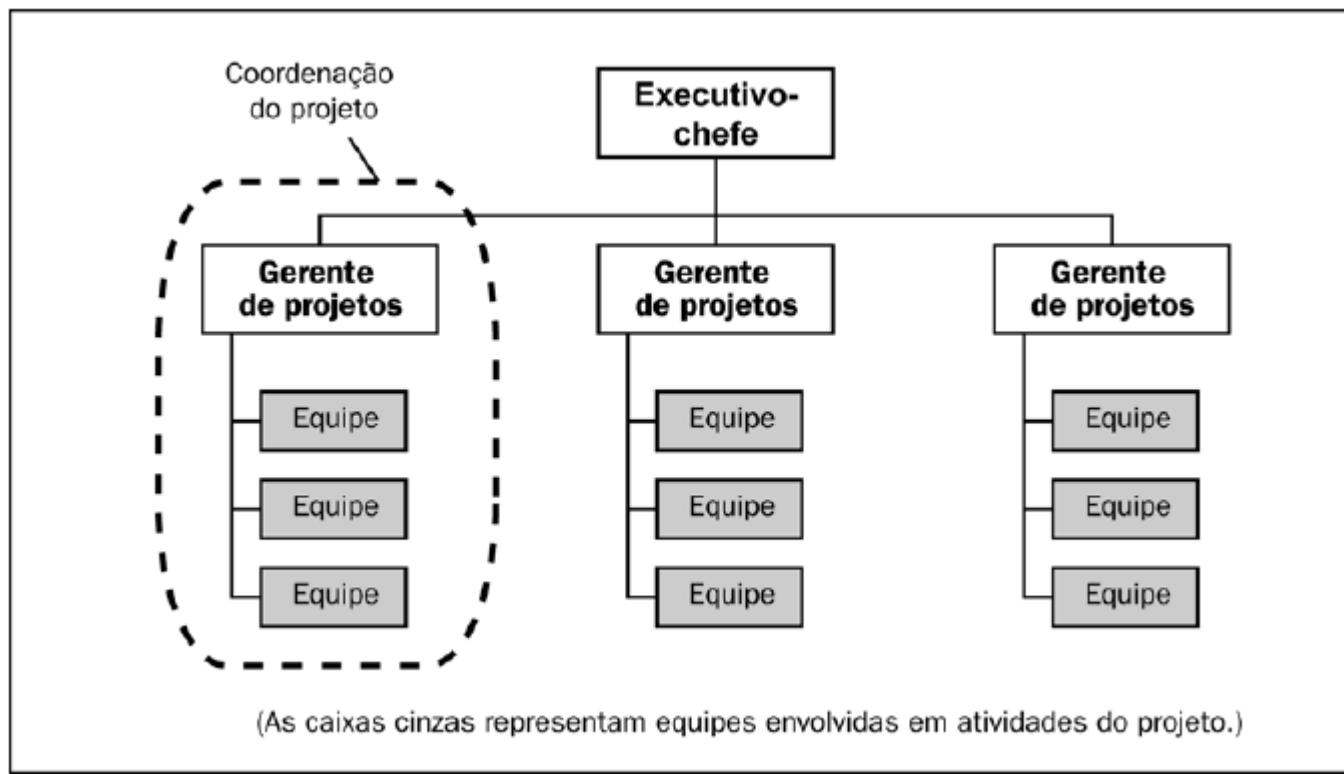
PLANEJAMENTO DO PROJETO - EQUIPE

○ Organização Funcional



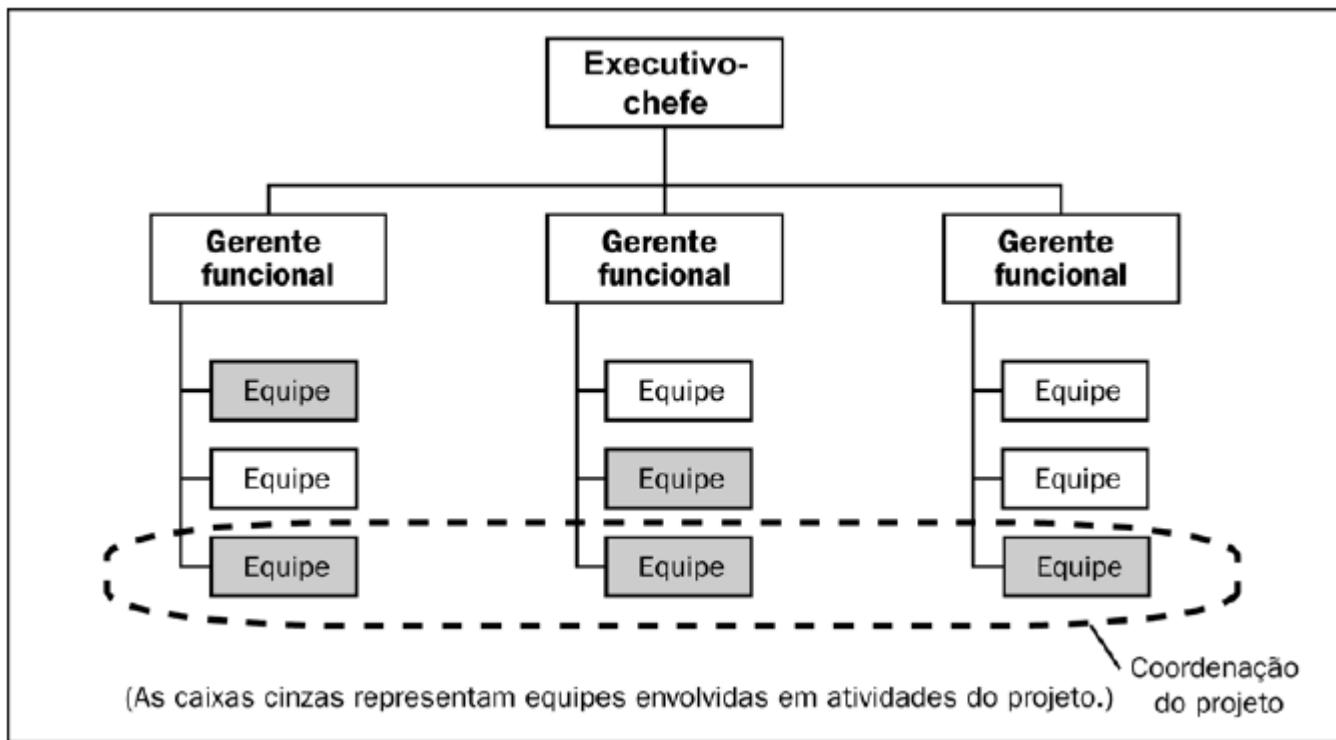
PLANEJAMENTO DO PROJETO - EQUIPE

- Estrutura projetizada



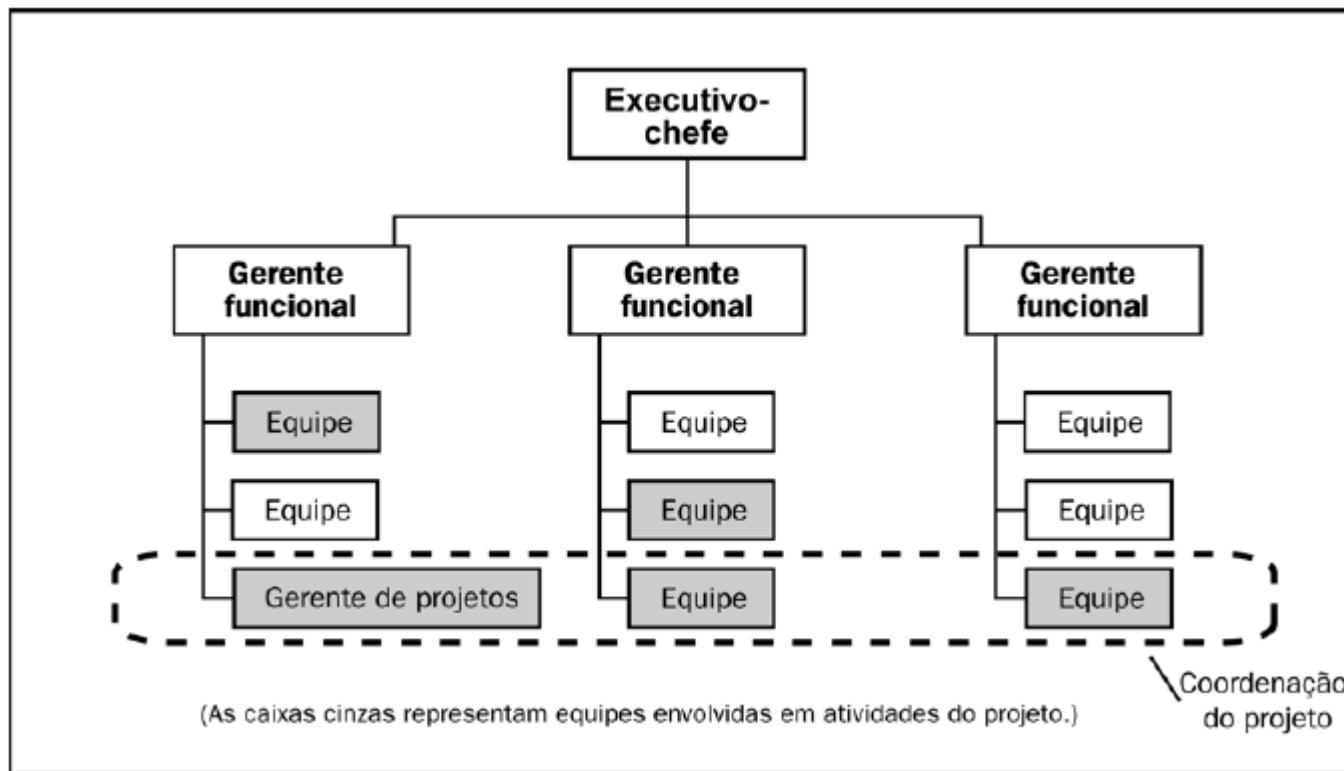
PLANEJAMENTO DO PROJETO - EQUIPE

○ Estrutura matricial Fraca



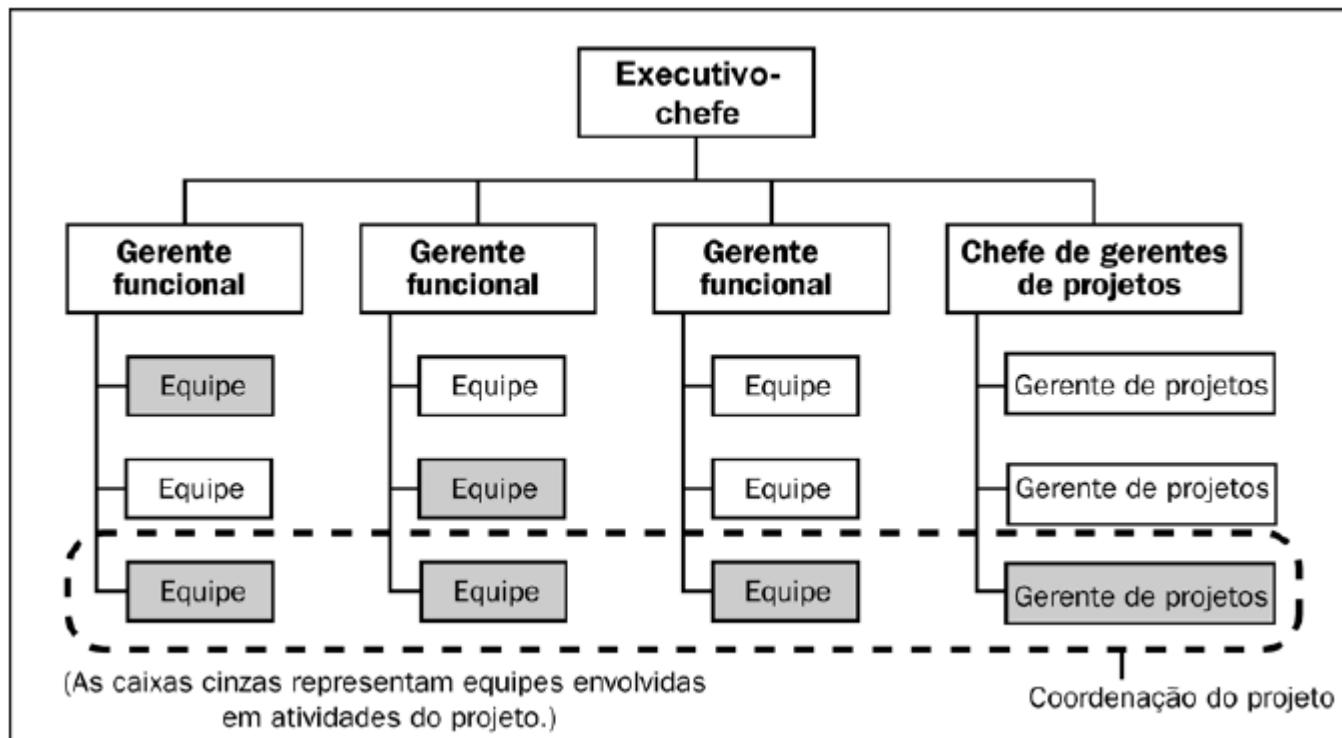
PLANEJAMENTO DO PROJETO - EQUIPE

- Estrutura matricial balanceada



PLANEJAMENTO DO PROJETO - EQUIPE

- Estrutura matricial forte



PLANEJAMENTO DO PROJETO - EQUIPE

- Vantagens da estrutura matricial
 - Objetivo do projeto visível;
 - Redução de custos com o rateio das áreas administrativas;
 - Resposta rápida para demandas da empresas;
 - Maior eficiência no uso dos recursos;
 - Coordenação de esforços otimizada nas gerências funcionais;
 - Encerramento do projeto menos traumático;
 - Acesso a especialistas.

PLANEJAMENTO DO PROJETO - EQUIPE

- Desvantagens da estrutura matricial
 - Equipe do projeto reportando para duas pessoas;
 - Maior complexidade da gestão das pessoas;
 - Dificuldade na alocação e priorização dos recursos;
 - Diferentes prioridades entre os gerentes;
 - Custo com a definição de processos e procedimentos;
 - Conflitos com a disputa de poder.

PLANEJAMENTO DO PROJETO - EQUIPE

○ Estruturas organizacionais

Características do projeto	Estrutura da organização	Funcional	Matricial			Por projeto
			Fraca	Balanceada	Forte	
Autoridade do gerente de projetos	Pouca ou nenhuma	Limitada	Baixa a moderada	Moderada a alta	Alta a quase total	
Disponibilidade de recursos	Pouca ou nenhuma	Limitada	Baixa a moderada	Moderada a alta	Alta a quase total	
Quem controla o orçamento do projeto	Gerente funcional	Gerente funcional	Misto	Gerente de projetos	Gerente de projetos	
Função do gerente de projetos	Tempo parcial	Tempo parcial	Tempo integral	Tempo integral	Tempo integral	
Equipe administrativa do gerenciamento de projetos	Tempo parcial	Tempo parcial	Tempo parcial	Tempo integral	Tempo integral	

C - Coordena (lidera/executa)**P** - aProva**A** - Apoia (atua sempre)**S** - atua quando Solticado**Q**- arQuiva**T** - Todos atuam, responde o diretor

Presidência	Vice Presidencia	Diretoria de Certificação	Diretoria de Filiação	Diretoria de Finanças	Diretoria de Eventos	Diretoria de Marketing	Diretoria de Operações	Diretor Ex-Ofício	Conselho de Orientação	Administrativo	Contador	Webmaster
-------------	------------------	---------------------------	-----------------------	-----------------------	----------------------	------------------------	------------------------	-------------------	------------------------	----------------	----------	-----------

Relações Institucionais

Relações com o PMI: Cartas e e-mails	C	A	S	S	S	S	S	S	S	A	-	S
Entrega de Relatórios ao PMI	C	A	S	S	S	S	S	S	S	A	-	-
Participação em Eventos do PMI	S	S	S	S	S	S	S	S	-	-	S	-
Realizaccao de Palestras Institucionais	S	S	S	A	S	S	C	A	-	-	A	-

Finanças

Receitas Relativas a anuidades (PMI)	S	S	A	A	C	A	-	-	-	A	S	-
Receitas de Eventos	-	-	-	A	C	A	-	-	-	A	S	-
Contabilidade do PMI-PE	P	A	-	-	C	-	-	S	-	-	A	S
Orçamento Anual do PMI-PE	P	A	S	S	C	S	S	A	-	-	A	A

Site PMI-PE

WEB Matérias Técnicas	P	A	A	A	A	A	C	A	A	A	A	-
WEB Divulgação de Eventos PMI-PE	S	S	A	A	S	P	C	A	-	-	A	S

Educação

Grupos de Estudos Preparatórios	S	S	C	S	S	A	A	A	-	-	A	-
Cursos	S	S	C	S	A	A	A	A	-	-	A	-
Apoio a Exames/Manutenção de Certificação	S	S	C	A	S	S	S	S	-	-	-	S
Emissão de Certificados	S	S	C	A	S	S	S	S	-	-	A	-

Eventos

Programação Anual de Eventos	P	A	S	S	A	C	S	A	S	S	A	-
Organização de Eventos	P	A	S	S	A	C	A	A	S	S	A	-

Apoio aos Membros

Controle de acesso ao DEP	P	-	-	-	-	-	-	-	-	-	-	-
Retenção de Filiados	P	-	-	C	-	-	A	A	-	-	A	-
Coordenação de Voluntariado	A	A	A	A	A	A	A	C	-	-	S	-

Solicitações Gerais do Público

Inscrição como Membros no PMI/PMI-PE	S	S	-	C	A	-	-	-	-	A	-	-
Informações sobre o PMI/PMI-PE	A	S	S	C	-	S	S	S	-	-	A	-
Oportunidades Profissionais	A	A	A	A	A	A	C	A	-	-	A	-
Reunião Semanal	C	T	T	T	T	T	T	T	S	S	S	-

MATRIZ DE RESPONSABILIDADES

Projeto Construindo um Sorriso**MATRIZ DE
RESPONSABILIDADES**

Criado por Paulo Henrique Ladeira

ATIVIDADE / DOCUMENTO	PARTICIPANTES				
	Gerente do projeto (Luiz)	Gerente da qualidade (Ricardo)	Gerente de planejamento (Christian)	Gerente de compras (Claudio)	Gerente Financeiro (Rogério)
1 Escopo	E	E	E	E	E
2 Documentação do projeto	E	I	E	A	I
3 Elaboração da equipe	A	I	A	I	E
4 Comunicação	E	A	E	I	I
5 Plano de qualidade	A	E	A	I	I
6 Plano de gerenciamento de riscos	E	A	E	I	I
7 Planejamento	A	I	E	I	E
8 Obtenção dos recursos	E	E	E	E	E
9 Preparação do consultório	E	I	E	E	E
10 Aquisição	A	I	A	E	A
11 Entrega	F	F	F	F	F

LEGENDA	PARTICIPANTES		
E	EXECUTA	Gerente de Projetos	Luiz
A	APROVA	Gerente da Qualidade	Ricardo
F	FISCALIZA	Gerente de Planejamento	Christian
I	RECEBE PARA INFORMAÇÃO	Gerente de Compras	Claudio
		Gerente Financeiro	Rogério

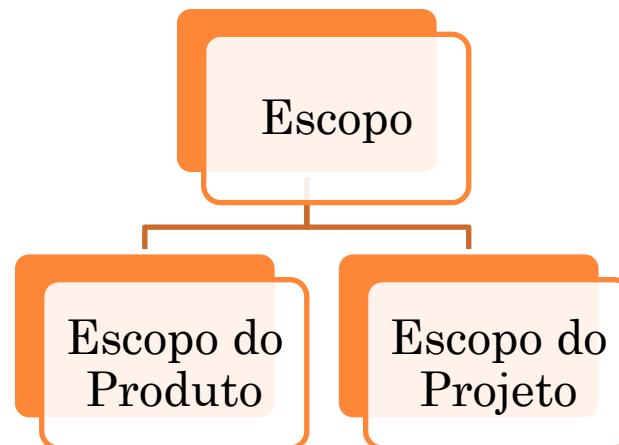
PLANEJAMENTO DO PROJETO - EQUIPE



Lição de Gerenciamento: Nunca comece um projeto sem ter em mãos todos os recursos.

PLANEJAMENTO DO PROJETO - ESCOPO

- Necessário limitar o escopo do produto e do projeto.
- Escopo do produto deve ser mais forte que o do projeto.



PLANEJAMENTO DO PROJETO – DEFINIÇÃO DO ESCOPO

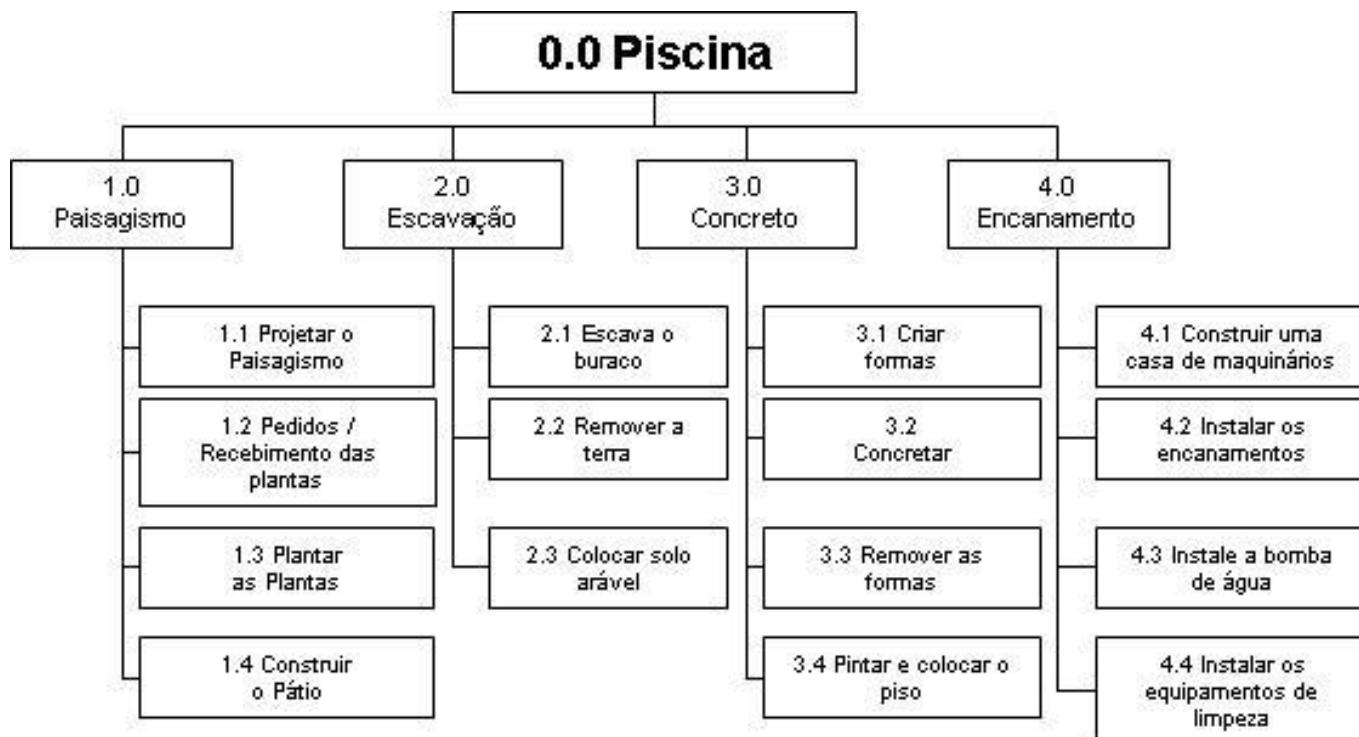
- Pré-requisito para o sucesso do projeto;
- Deve conter a visão de todas as partes interessadas;
- Todas as necessidades e desejos devem ser convertidas para requisitos;
- Requisitos devem ser alinhados com as metas do projeto.
- Devemos contemplar:
 - O que deve ser feito;
 - O que não deve ser feito;
 - Produtos e especificações.

PLANEJAMENTO DO PROJETO – DEFINIÇÃO DA EAP

- Estrutura analítica do projeto (EAP) ou Work breakdown structure (WBS)
- Subdivide o produto do projeto em partes menores e mais facilmente gerenciáveis (guiados por entregas e não ações)
- Identifica todo o trabalho a ser executado (regra dos 100%)
- Define a linha de base do escopo do projeto

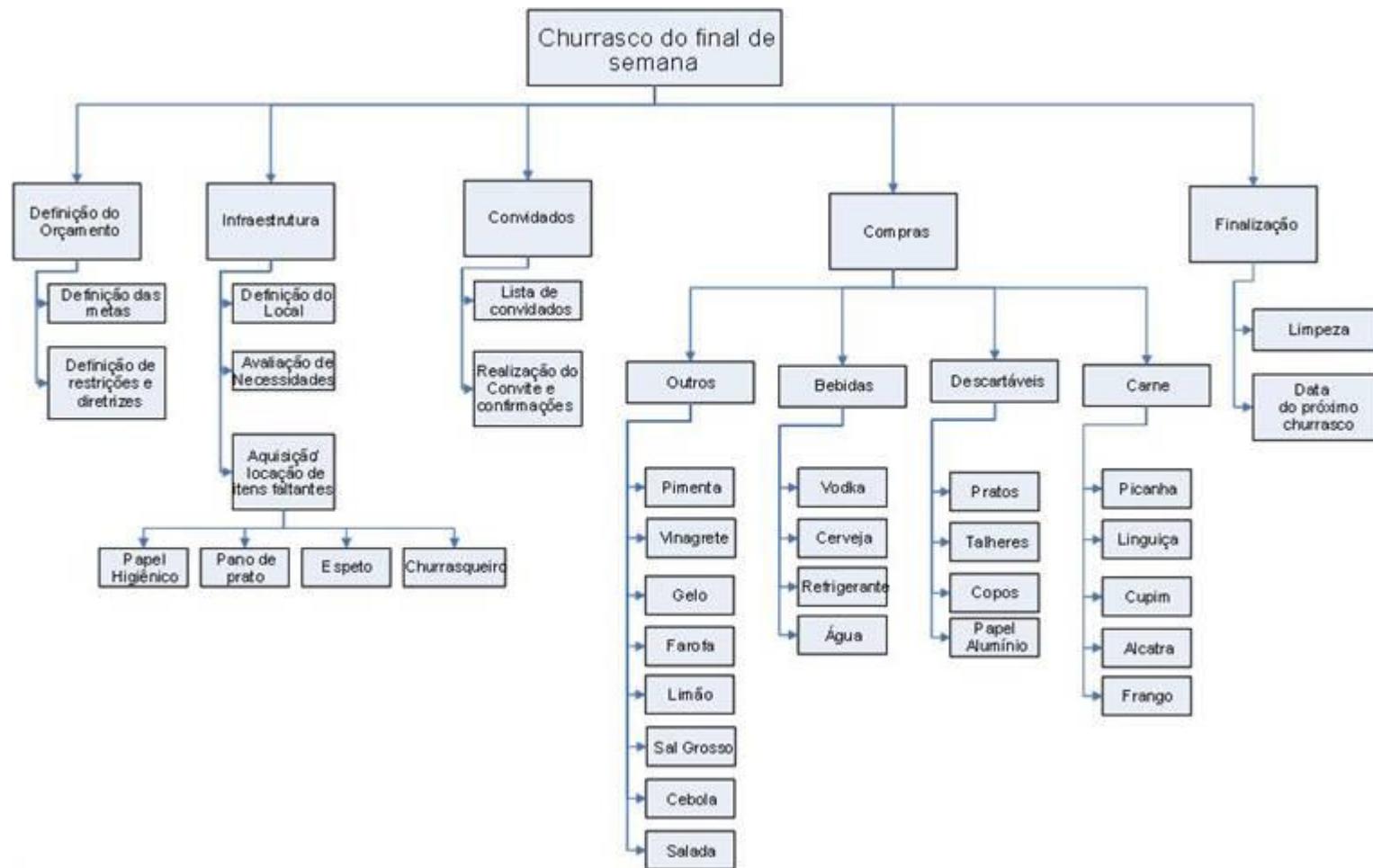
PLANEJAMENTO DO PROJETO – DEFINIÇÃO DA EAP

○ Projeto Piscina



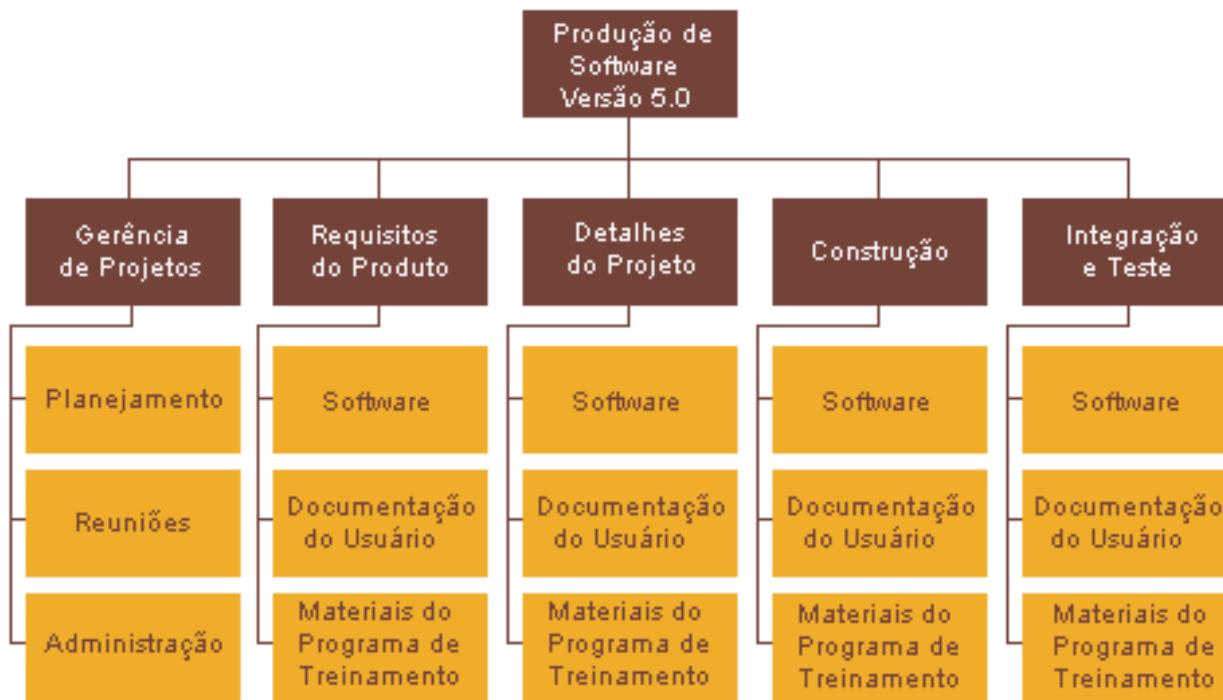
PLANEJAMENTO DO PROJETO – DEFINIÇÃO DA EAP

○ Projeto Churrasco



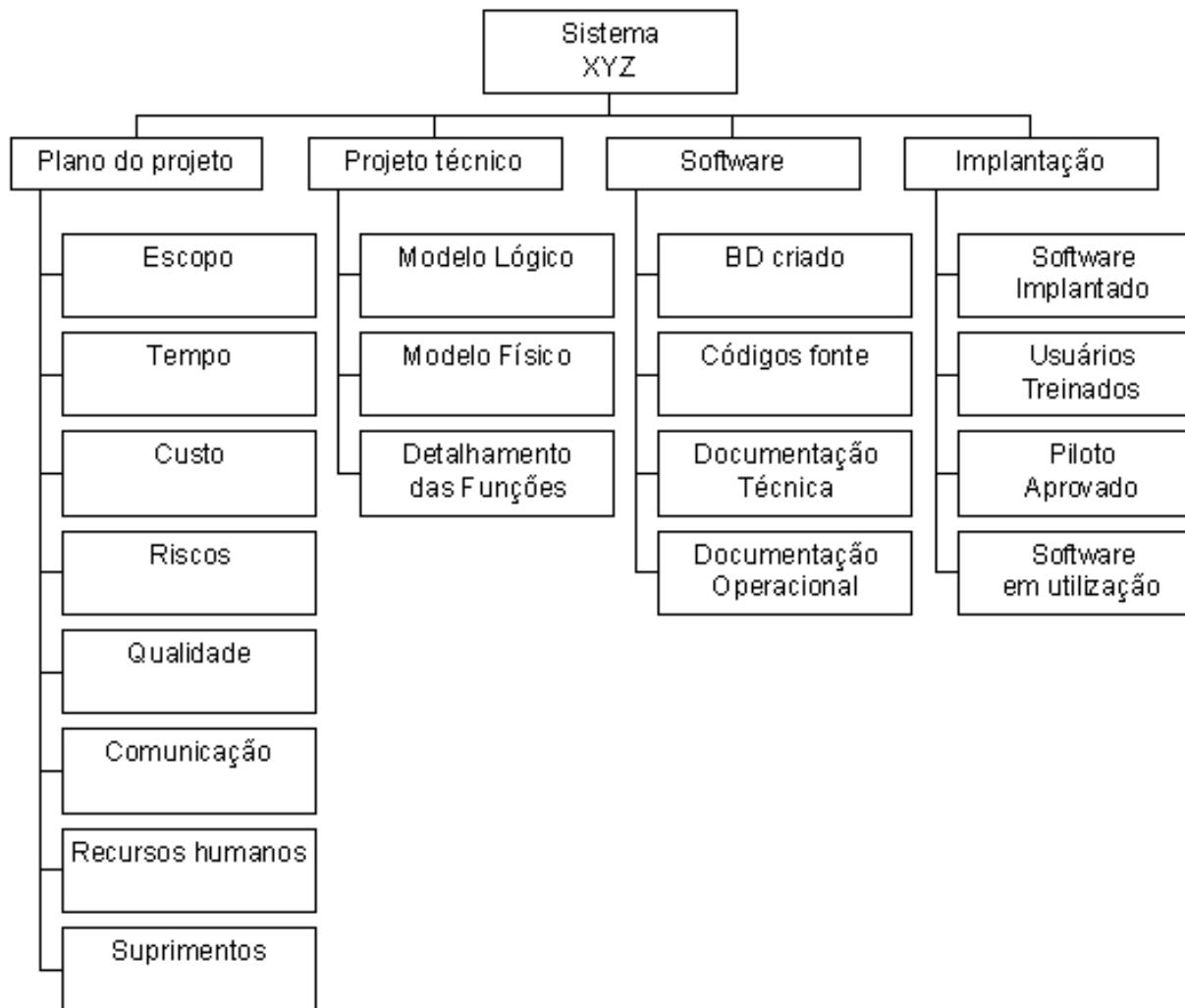
PLANEJAMENTO DO PROJETO – DEFINIÇÃO DA EAP

- EAP por fase



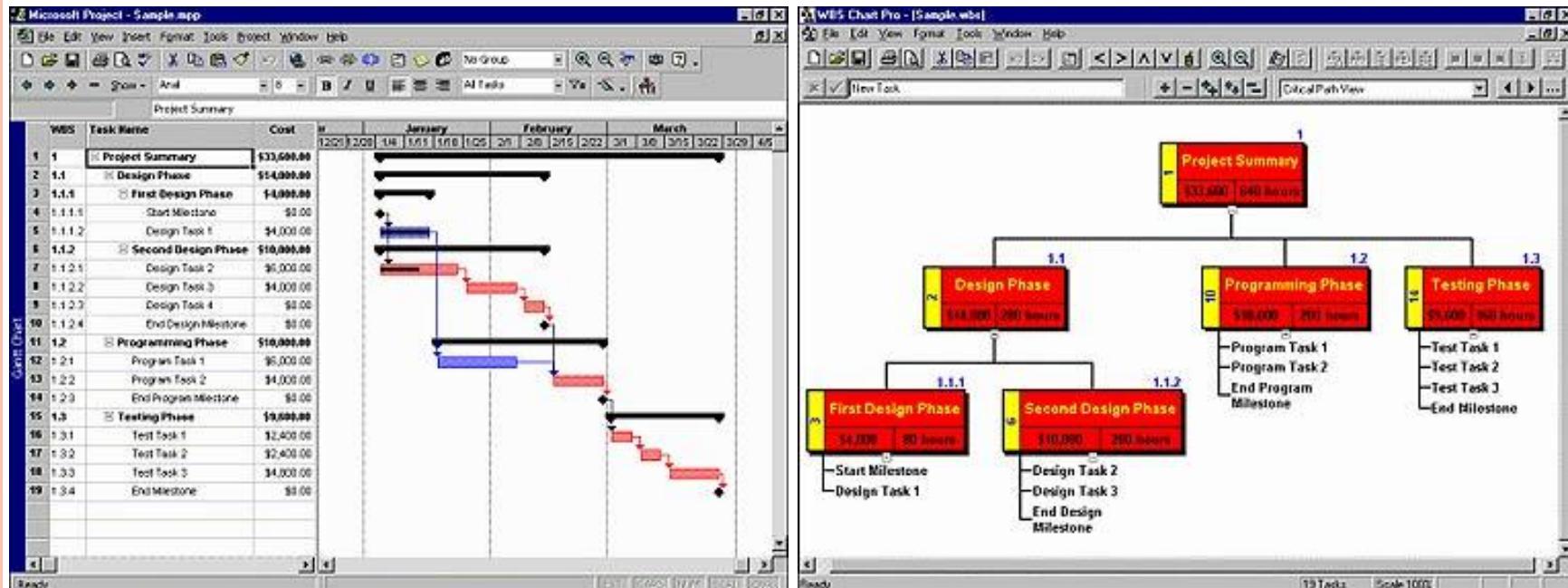
PLANEJAMENTO DO PROJETO – DEFINIÇÃO DA EAP

○ EAP de software



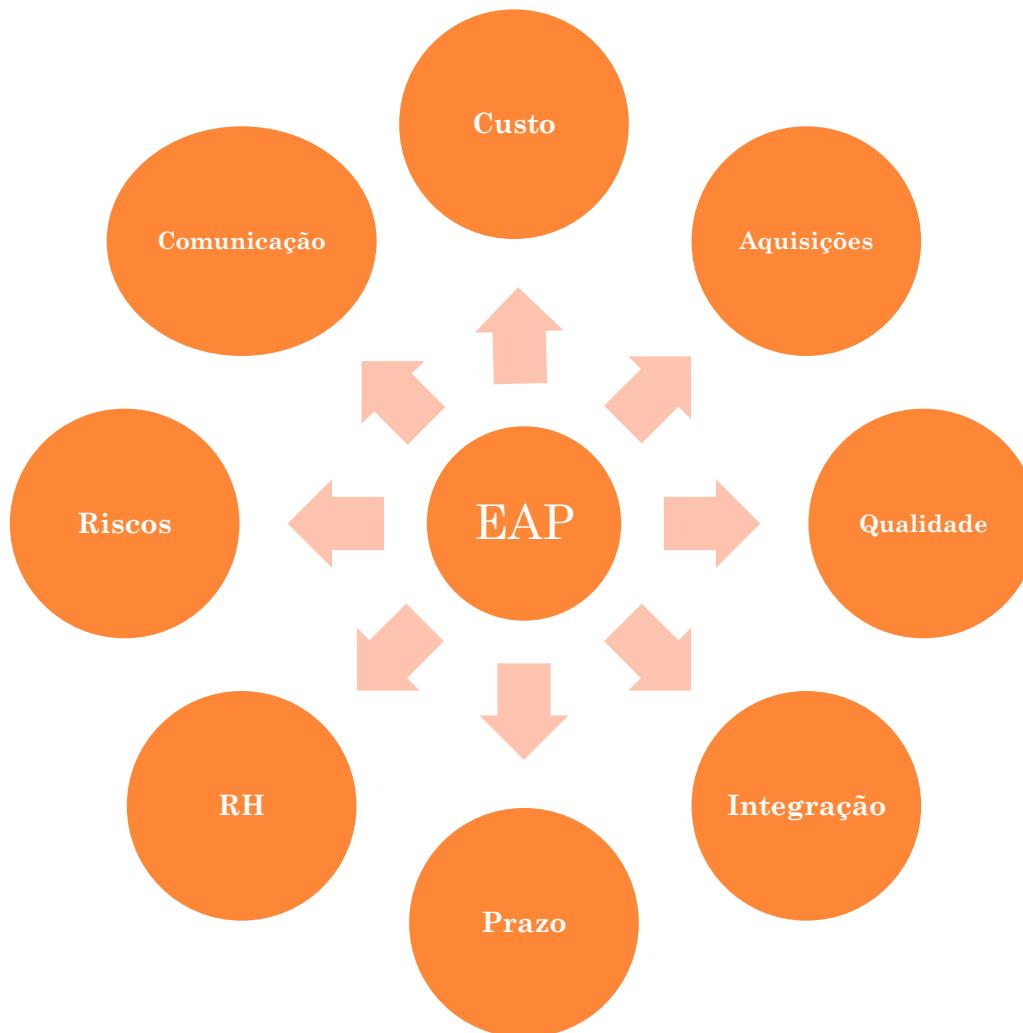
PLANEJAMENTO DO PROJETO – DEFINIÇÃO DA EAP

○ Integração Project e WBS Chart Pro



Criado por Paulo Henrique Ladeira

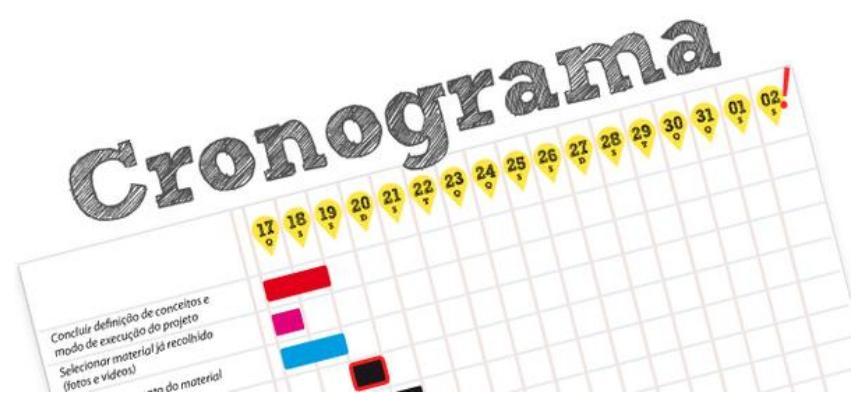
PLANEJAMENTO DO PROJETO – DEFINIÇÃO DA EAP



Criado por Paulo Henrique Ladeira

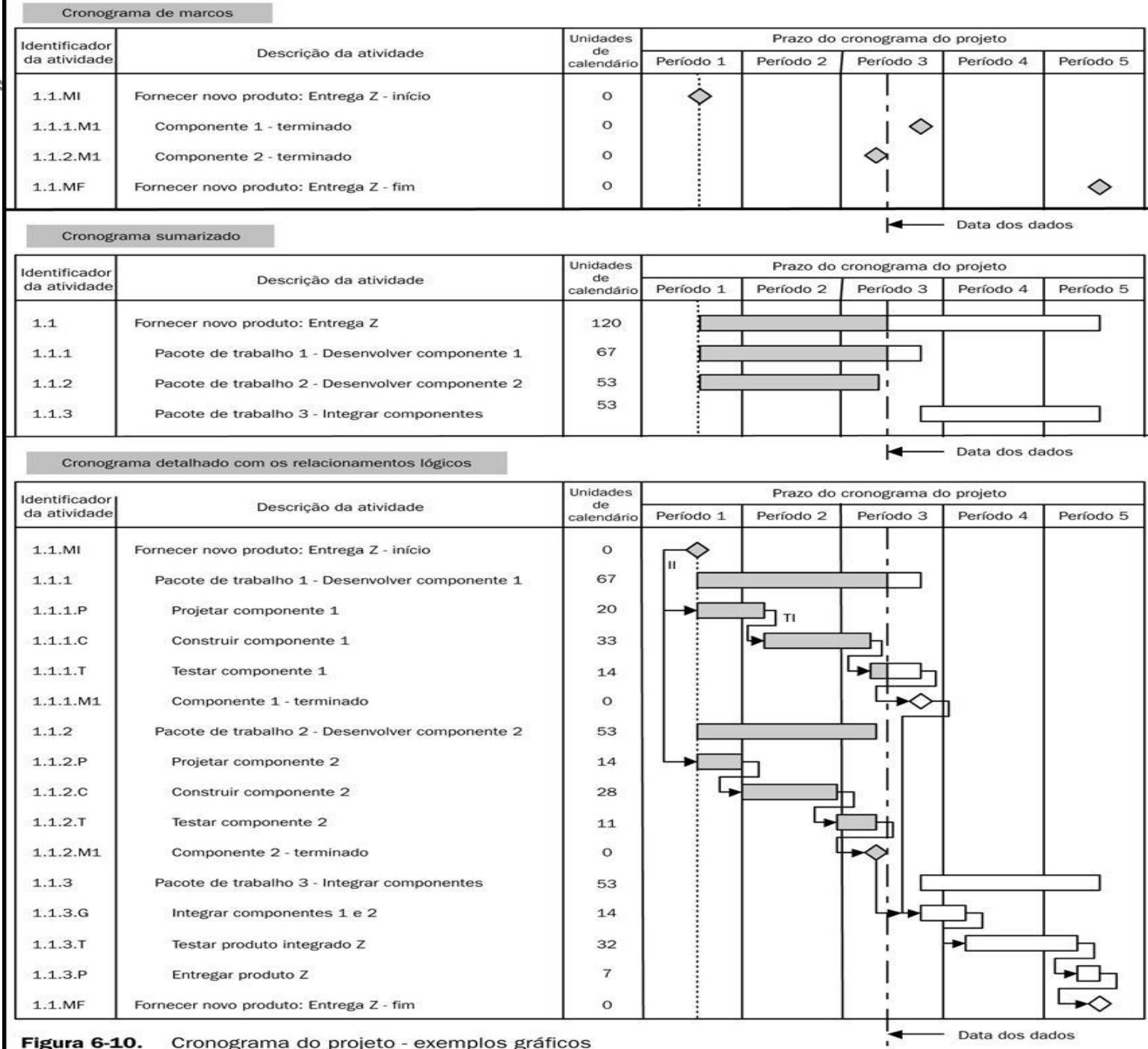
PLANEJAMENTO DO PROJETO - CRONOGRAMA

- Nasce a partir da EAP
- Para construir um cronograma, torna-se necessário conhecer:
 - Escopo do trabalho;
 - Tipos de recursos;
 - Produtividade dos recursos;
 - Disponibilidade dos recursos;
 - Calendário dos recursos.

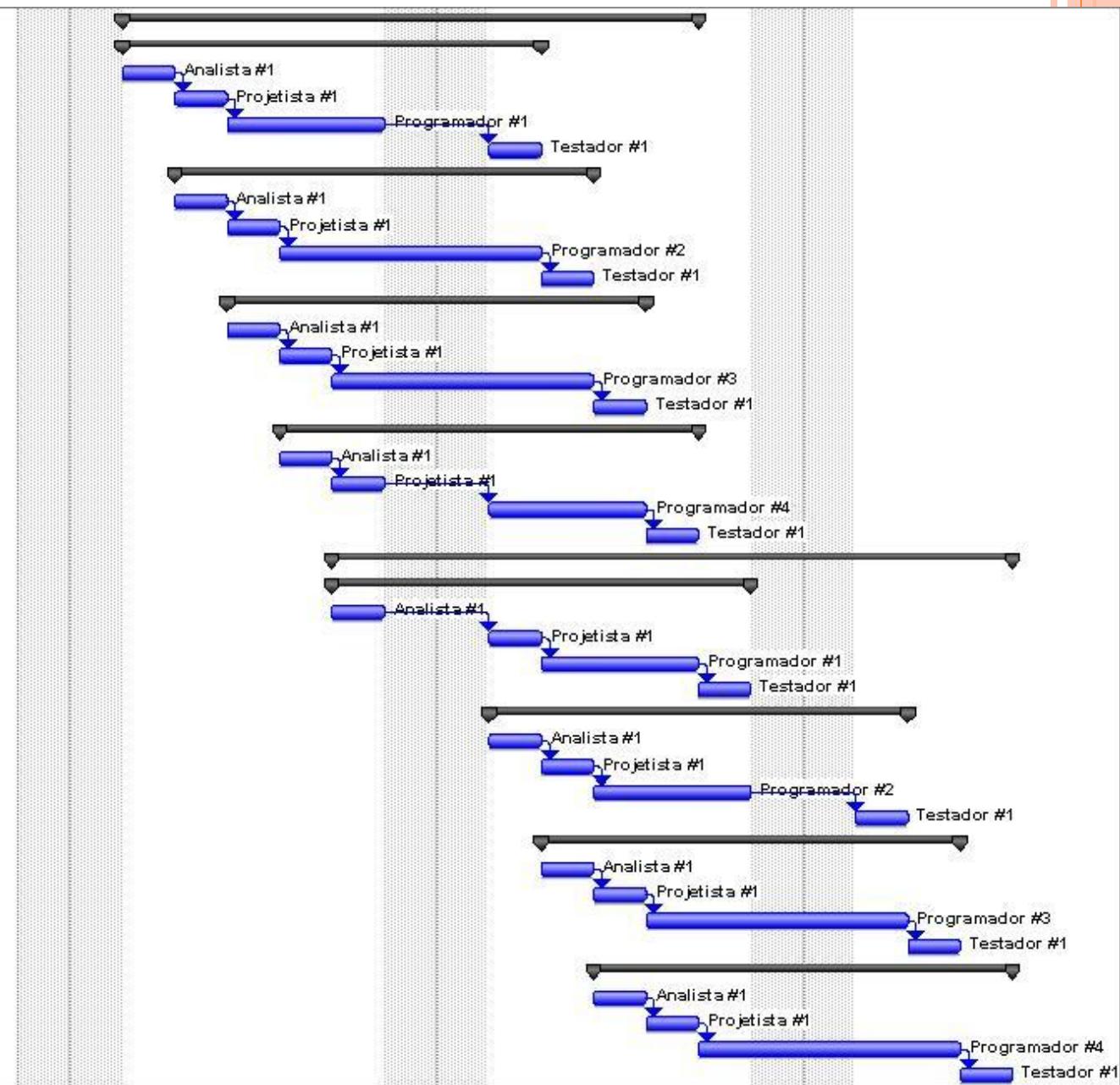


PLANEJAMENTO DO PROJETO - CRONOGRAMA

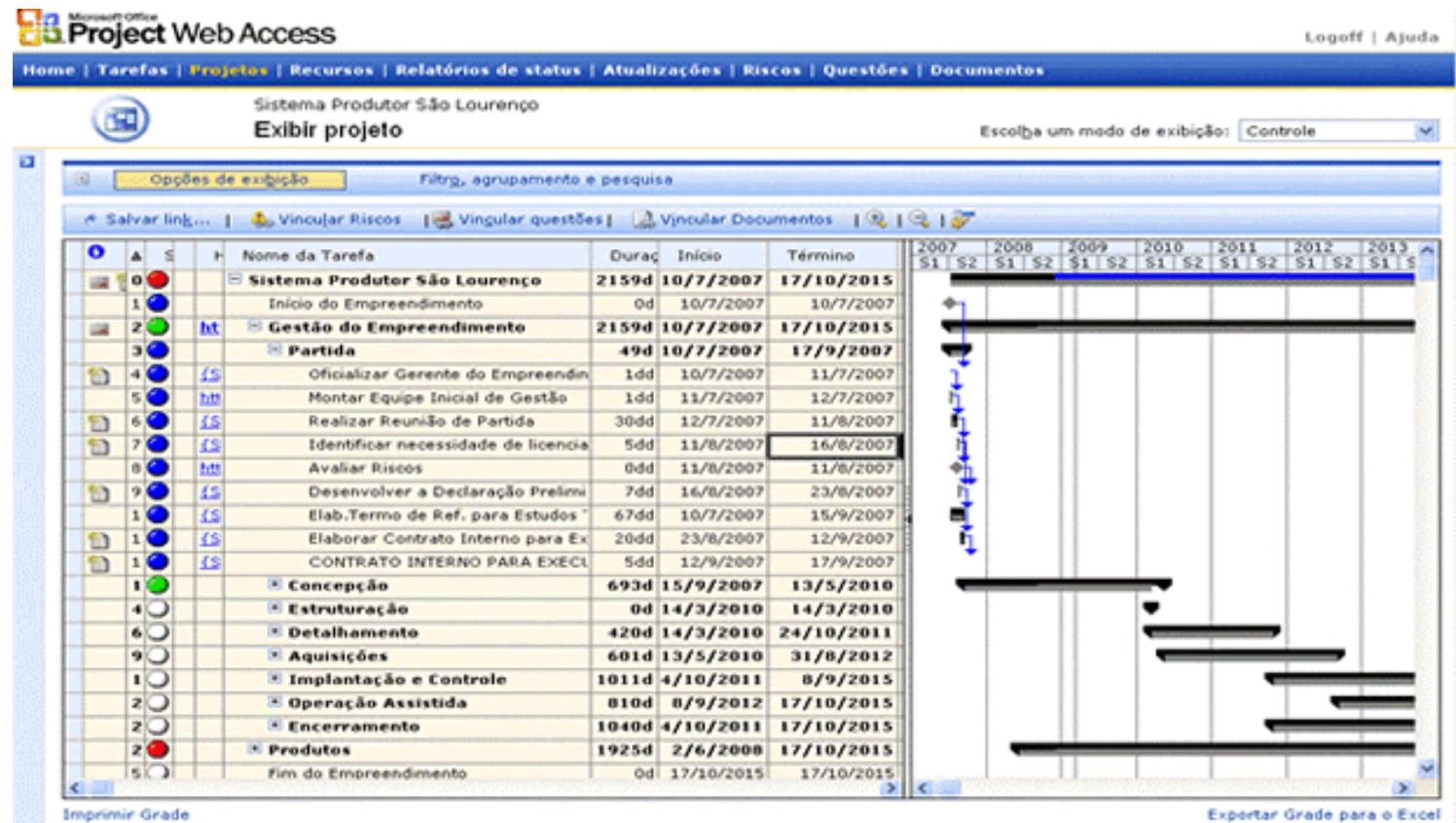
- Determina as datas de início e término, junto com as durações das atividades/tarefas.
- Geralmente, leva a necessidade de revermos as estimativas.
- O cronograma é continuado durante todo o projeto.
- Após criado, deve ser aceito e validado pelos stakeholders.
- Define a linha de base de prazo do projeto.
- Deve conter o cronograma detalhado e o cronograma de marcos do projeto.



Release #1	
Funcionalidade #1	
Especificação	Analista #1
Projeto	Projetista #1
Programação	Programador #1
Testes	Testador #1
Funcionalidade #2	
Especificação	Analista #1
Projeto	Projetista #1
Programação	Programador #2
Testes	Testador #1
Funcionalidade #3	
Especificação	Analista #1
Projeto	Projetista #1
Programação	Programador #3
Testes	Testador #1
Funcionalidade #4	
Especificação	Analista #1
Projeto	Projetista #1
Programação	Programador #4
Testes	Testador #1
Release #2	
Funcionalidade #5	
Especificação	Analista #1
Projeto	Projetista #1
Programação	Programador #1
Testes	Testador #1
Funcionalidade #6	
Especificação	Analista #1
Projeto	Projetista #1
Programação	Programador #2
Testes	Testador #1
Funcionalidade #7	
Especificação	Analista #1
Projeto	Projetista #1
Programação	Programador #3
Testes	Testador #1
Funcionalidade #8	
Especificação	Analista #1
Projeto	Projetista #1
Programação	Programador #4
Testes	Testador #1

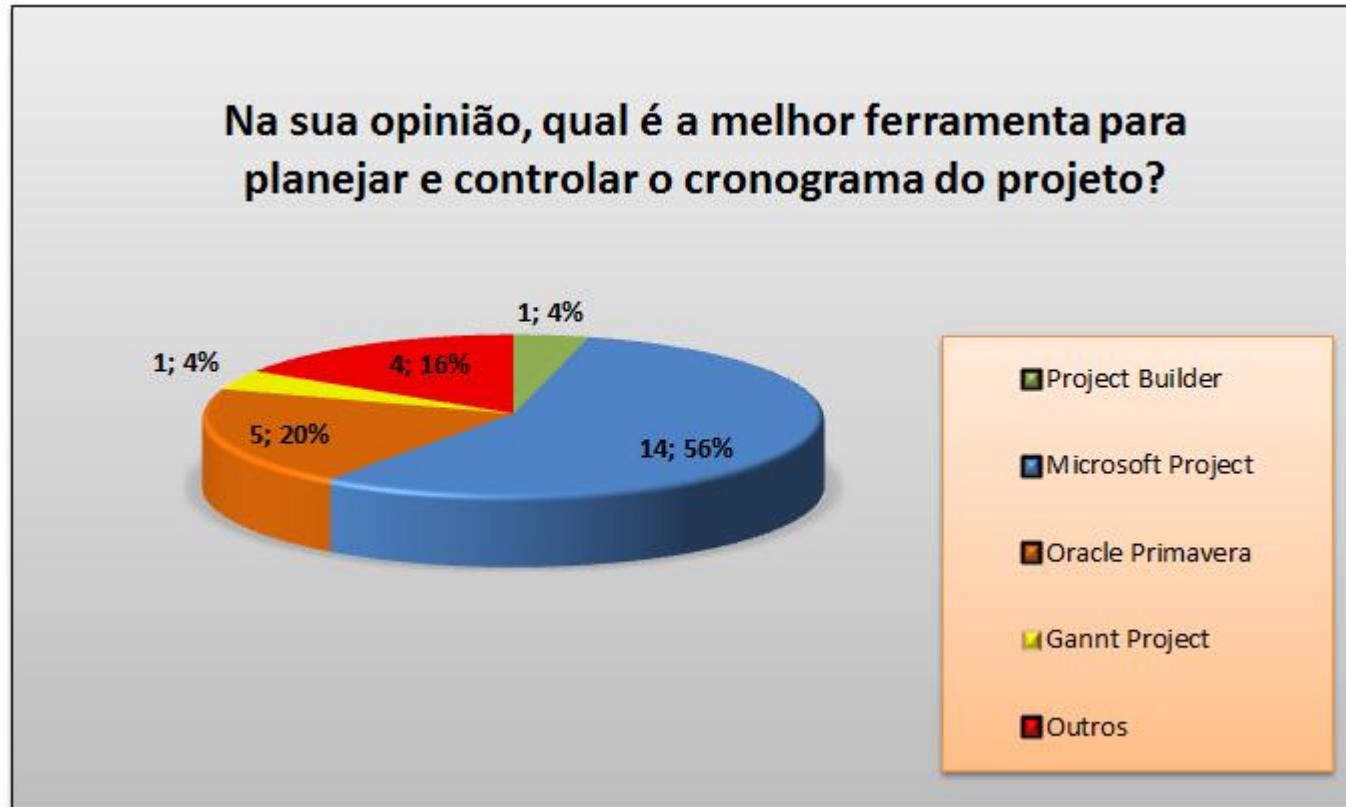


PLANEJAMENTO DO PROJETO - CRONOGRAMA



Criado por Paulo Henrique Ladeira

PLANEJAMENTO DO PROJETO - CRONOGRAMA



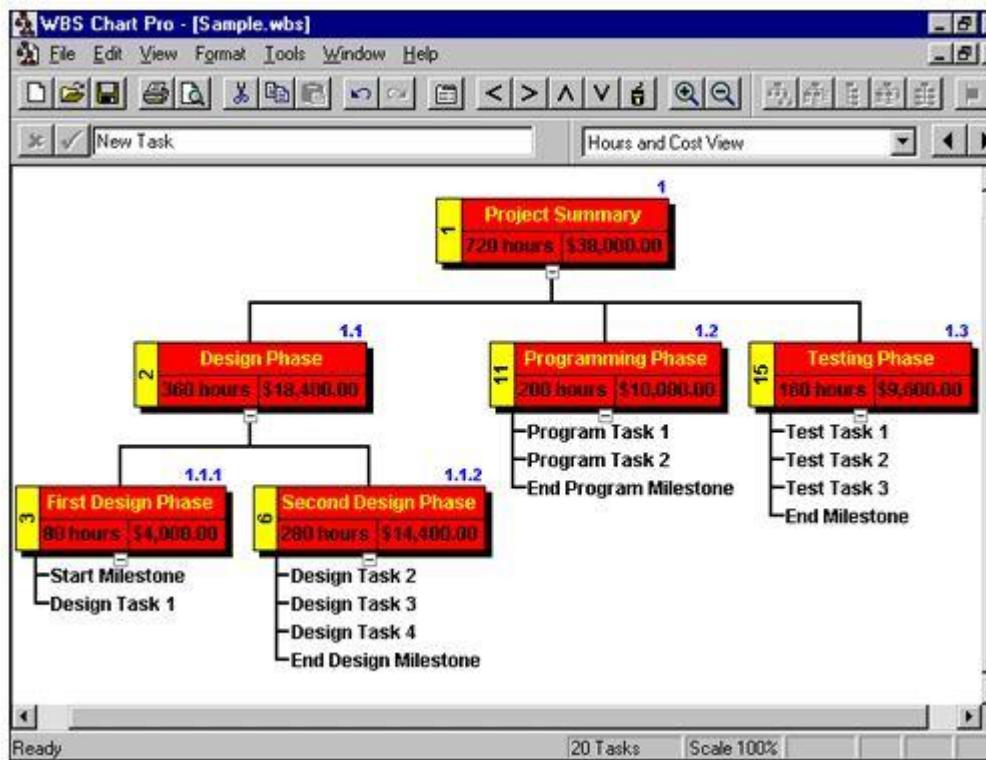
- Fonte: <http://danielettinger.com/2011/07/05/resultado-da-enquete-%E2%80%9Cna-sua-opiniao-qual-e-a-melhor-ferramenta-para-planejar-e-controlar-o-cronograma-do-projeto/>

PLANEJAMENTO DO PROJETO - ORÇAMENTO

- Tem como objetivo definir os custos dos recursos (contratos) necessários ao projeto.
- Diversas variáveis devem ser consideradas, como aumento salarial, regras do sindicato, inflação, variação cambial, riscos, custos da organização, etc.
- As estimativas vão aumentando a assertividade com o andamento do projeto:
 - Iniciação: variação de -25% a +50%
 - Planejamento preliminar: variação de -10% a +25%
 - Planejamento detalhado: variação de -5% a +5%
- Define a linha de base de custos do projeto.

PLANEJAMENTO DO PROJETO - ORÇAMENTO

- Estimativa bottom-up



PLANEJAMENTO DO PROJETO - COMUNICAÇÃO

- Busca determinar as necessidades de informações e as comunicações com os stakeholders.
 - O que
 - Para quem
 - Quando
 - Como
 - Por quem
- Define **quais** stakeholders do projeto necessitam de **quais** informações e **quando** e **como** essas serão fornecidas.

PLANEJAMENTO DO PROJETO - COMUNICAÇÃO

- Diagrama de relação, interesse e poder na comunicação



PLANEJAMENTO DO PROJETO - COMUNICAÇÃO

Matriz de comunicação

Tipo	Objetivo	Meio	Frequência	Audiência	Responsável	Produto
Kickoff do projeto	Apresentar para equipe e cliente o projeto. Revisar os objetivos, requisitos e compromissos.	Presencial	Única	Stakeholders e equipe do projeto	Gerente do projeto	Ata de reunião
Levantamento de dados	Realizar e documentar o detalhamento dos requisitos. Validar junto ao usuário garantindo o entendimento do escopo e obtendo o comprometimento do mesmo para o projeto	Presencial ou conferência	10 dias no início do projeto	Usuário e Stakeholders	Analista	Ata de levantamento
Status com equipe	Revisar o status das atividades com a equipe	Presencial	Semanal	Equipe do projeto	Gerente do projeto	Ata de reunião
Status com cliente	Revisar o status do projeto com os stakeholders verificando se o mesmo se encontra no prazo e custo. Tratar desvios e elementos que possam prejudicar o andamento do projeto	Presencial	Semanal	Stakeholders	Gerente do projeto	Ata de reunião
Desempenho do projeto	Apresentar de forma independente o desempenho do projeto e seus principais indicadores	e-mail	Semanal	PMO	Gerente do projeto	Status Report
Validações	Validar as principais entregas do projeto	e-mail	-	Stakeholders e usuários	Analista ou gerente do projeto	Aprovação

Criado por Paulo Henrique Ladeira

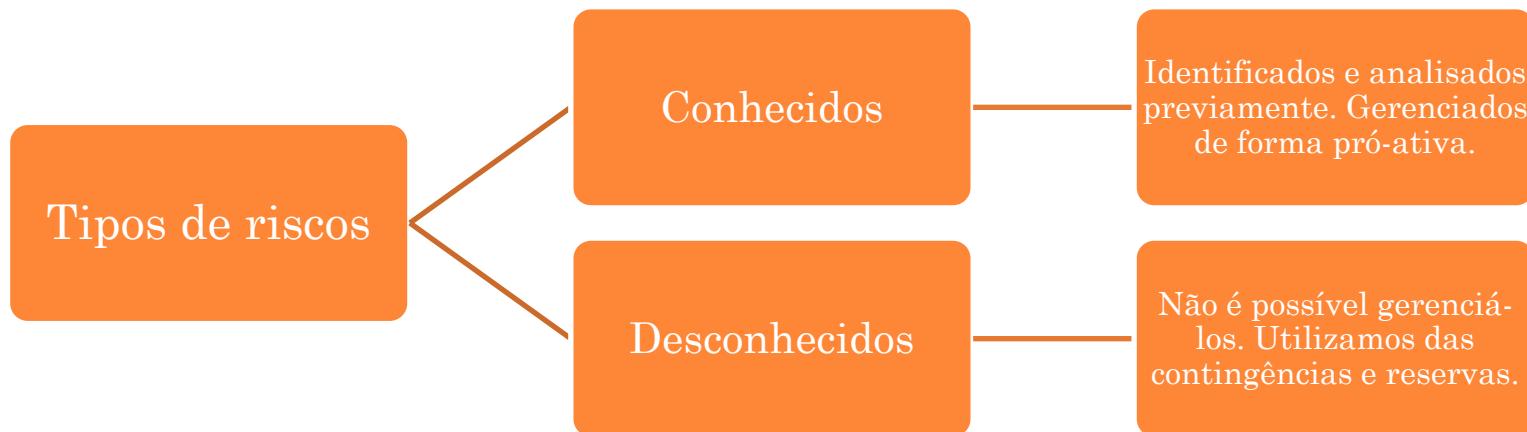
PLANEJAMENTO DO PROJETO - COMUNICAÇÃO

○ Matriz de comunicação

Tipo de Comunicação	Objetivo	Meio	Frequência	Audiência	Dono	Entregas	X
Reunião de início	Apresentar a equipe e o projeto. Revisar os objetivos do projeto e a abordagem de gestão	Face a Face	Uma vez	- Patrocinador - Time do projeto - Stakeholders	Gerente do Projeto	Ata da reunião	
Reuniões da equipe de projeto	Revisar o status do projeto com a equipe.	- Face a Face - Conferência telefônica	Semanalmente	Time do projeto	Gerente do Projeto	Ata da reunião	
Reuniões técnicas de design	Discutir e desenvolver o design técnico da solução do projeto.	Face a Face	Quando preciso	Pessoal técnico do projeto	Líder técnico	Ata da reunião	
Reuniões de status mensal do projeto	Relatar o status do projeto para a gestão.	- Face a Face - Conferência telefônica	Mensal	PMO	Gerente do Projeto		
Relatórios de status do projeto	Relatar o status do projeto, incluindo atividades, progresso, custos e problemas.	Email	Mensal	- Patrocinador - Time do projeto - Stakeholders PMO	Gerente do Projeto	Relatório de status do projeto	

PLANEJAMENTO DO PROJETO - RISCOS

- Busca identificar os riscos do projeto e planejar as ações.
- O risco é incerto e por isso é um risco. Se fosse fato que iria ocorrer, deixaria de ser um risco.

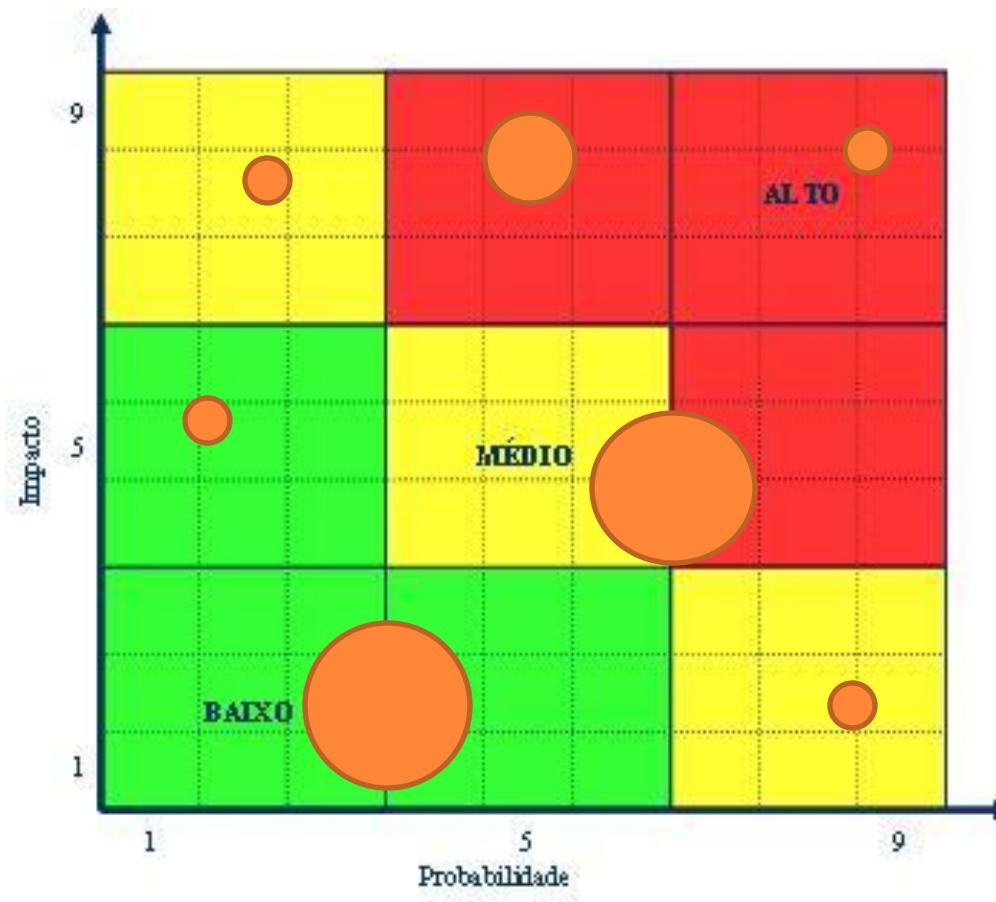


PLANEJAMENTO DO PROJETO - RISCOS

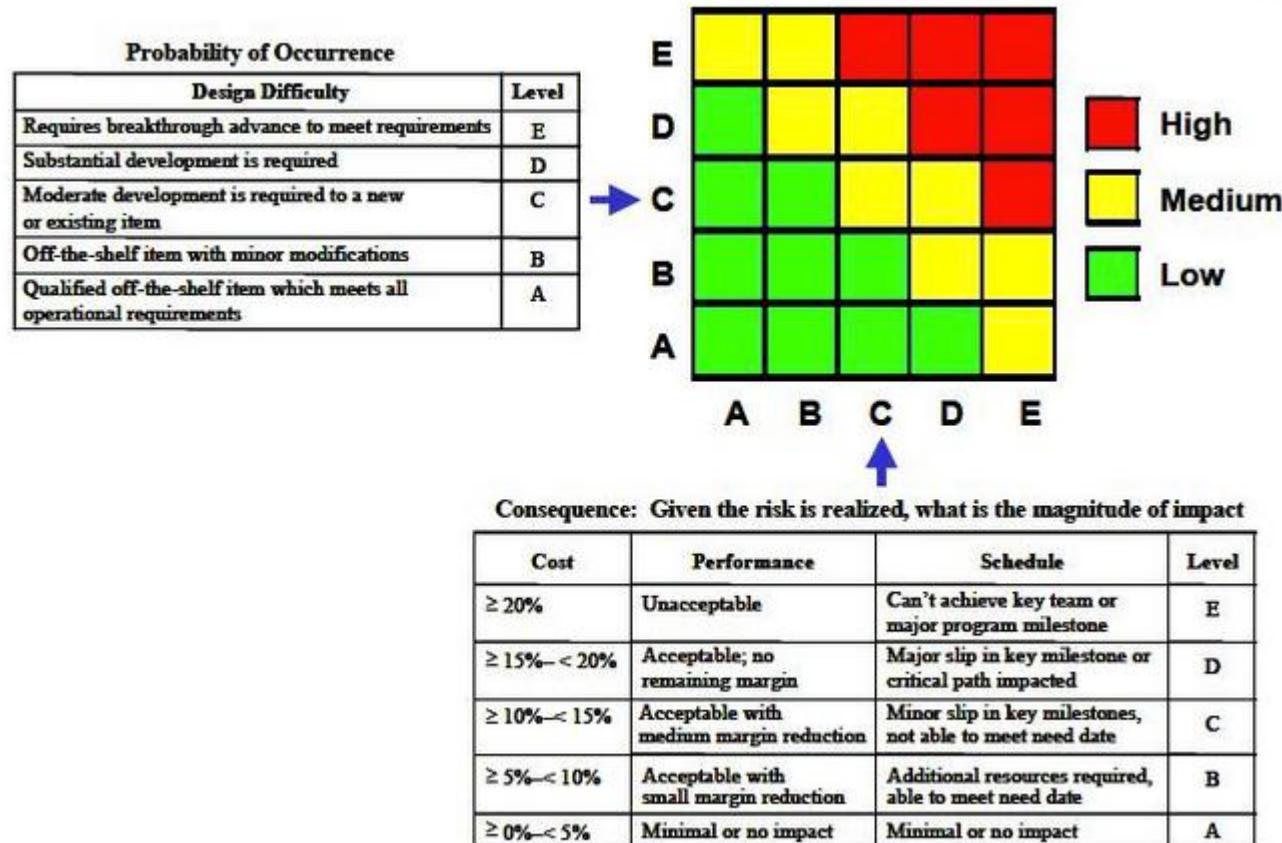
- Fontes de riscos em um projeto
 - Organizacionais (disponibilidade de pessoal habilitado)
 - Tecnologia (idade da tecnologia)
 - Fornecedores
 - Cliente
 - Externos (Cenário econômico)
 - Região (cultura, leis, etc.)
 - Escopo
 - Prazos
 - Disputas políticas
 - Conflitos internos
 - Metodologia não adequada
 - Etc.

PLANEJAMENTO DO PROJETO - RISCOS

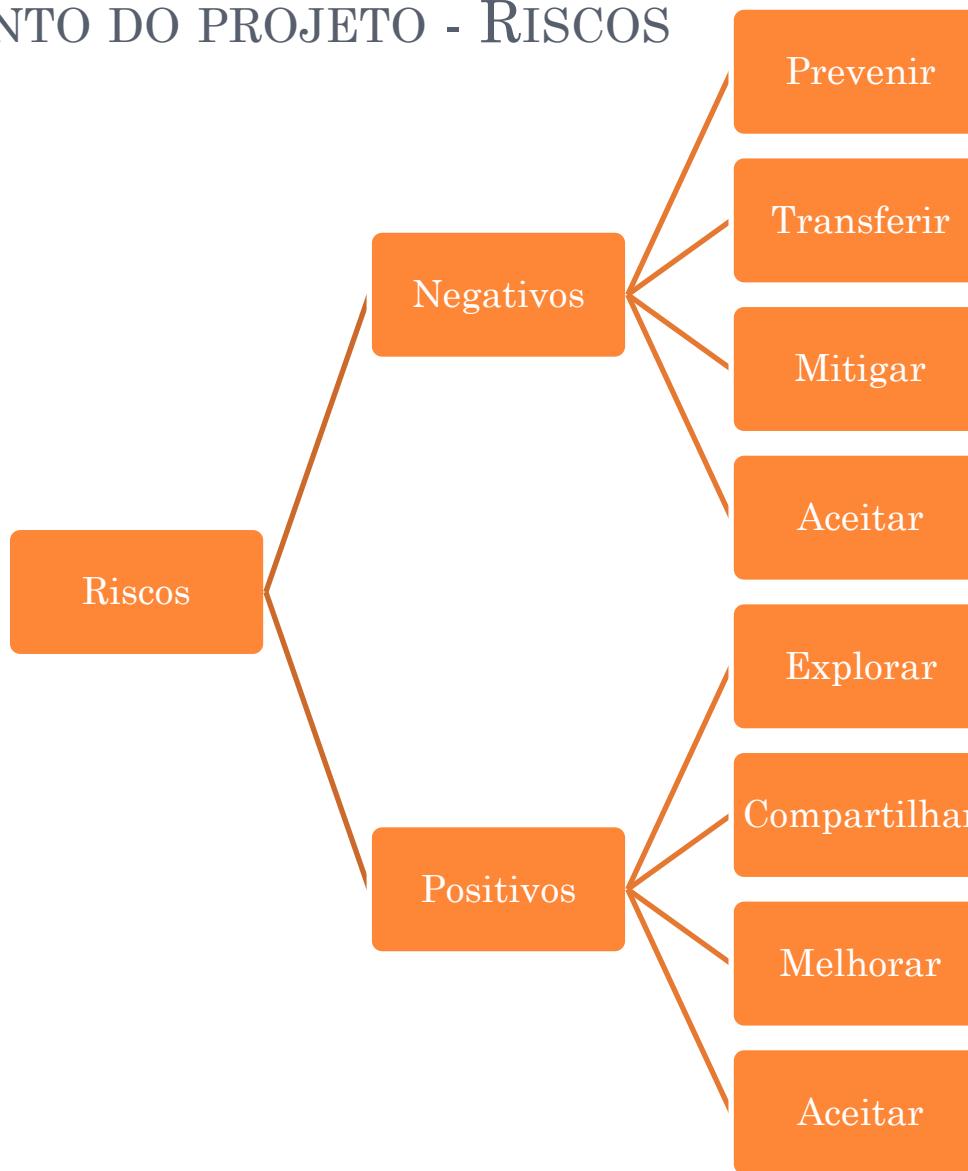
- Impacto x Probabilidade x Análise quantitativa



PLANEJAMENTO DO PROJETO - RISCOS



PLANEJAMENTO DO PROJETO - RISCOS



PLANEJAMENTO DO PROJETO - RISCOS

Criado por Paulo Henrique Ladeira



Riscos e gerenciamento de riscos são parte do jogo durante a execução de projetos.

GESTÃO DE PROJETOS

- Controlar e executar



CONTROLE DO PROJETO - MUDANÇAS



Not Invented Here™ © Bill Barnes & Paul Southworth

NotInventedHere.com

Criado por Paulo Henrique Ladeira

CONTROLE DO PROJETO - MUDANÇAS

- A única certeza que temos em projetos é que mudanças ocorrerão.
- Dos mais diversos tipos e em momentos e fases diferentes.
- Logo, a nossa única chance de sucesso é gerenciando as mudanças.

CONTROLE DO PROJETO - MUDANÇAS

○ Fontes de mudanças

- Escopo (incompleto e não detalhado)
- Alteração de requisitos (legais, ambientais, etc.)
- Qualidade
- Custos
- Prazos

CONTROLE DO PROJETO - MUDANÇAS

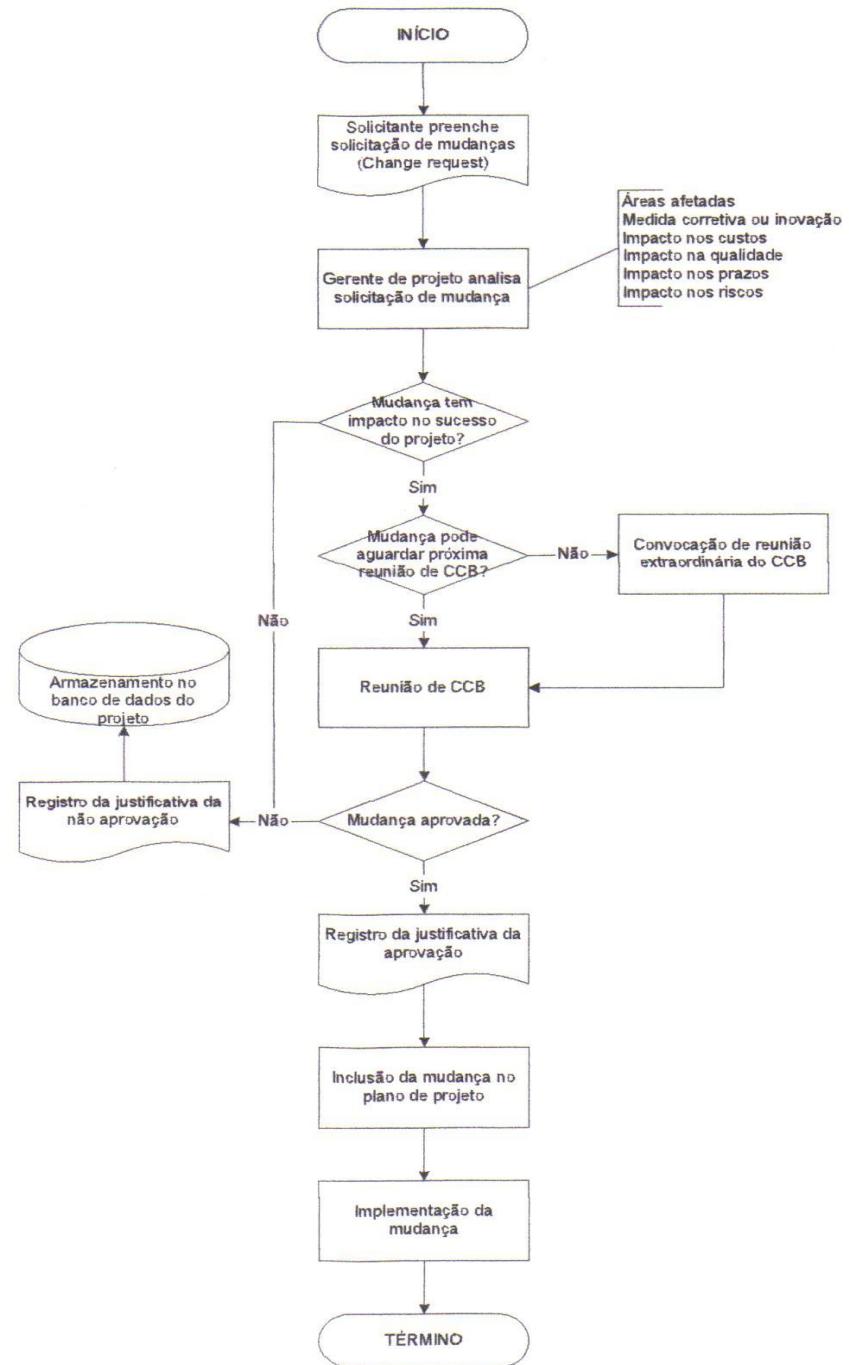
- Se não gerenciadas da forma correta, as mudanças podem causar:
 - Estouro no orçamento;
 - Atrasos no projeto;
 - Desvios em requisitos;
 - Conflitos e disputas;
 - Questionamentos legais;
 - Não aceitação do produto criado no projeto.

CONTROLE DO PROJETO - MUDANÇAS

- Mudanças bem gerenciáveis levam a:
 - Projetos bem-sucedidos;
 - Melhoria do ROI;
 - Satisfação do cliente e dos stakeholders.

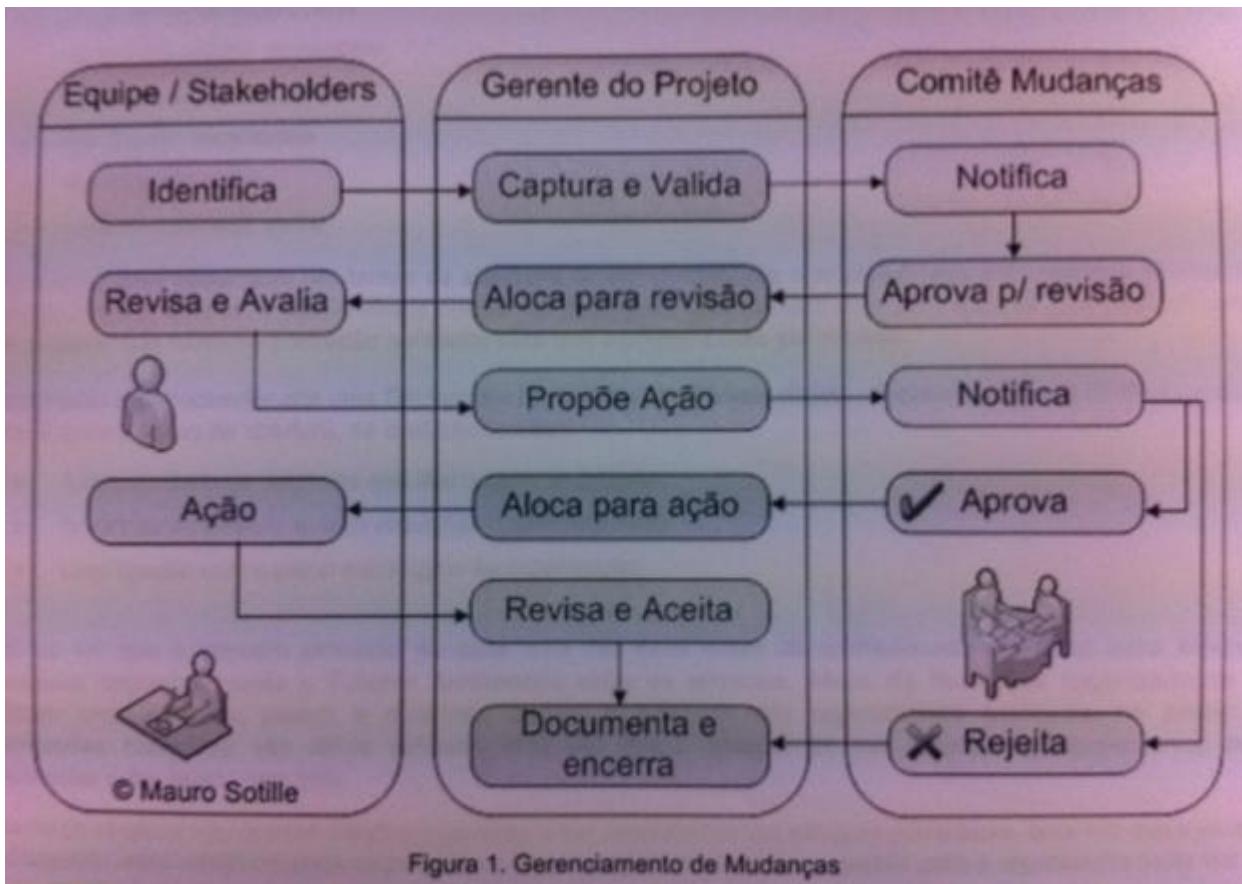
CONTROLE DO PROJETO

- Fluxo de controle integrado de mudanças



CONTROLE DO PROJETO - MUDANÇAS

- Fluxo de controle integrado de mudanças



CONTROLE DO PROJETO

- Processo dividido em:
 - Medir o desempenho real do projeto;
 - Comparar com o planejado;
 - Adotar ações corretivas, se necessário.

CONTROLE DO PROJETO

- Principais atividades:

- Monitorar desempenho do projeto periodicamente;
- Analisar os desvios entre previstos e realizados;
- Analisar as tendências;
- Planejar ações alternativas;
- Realizar simulações;
- Ajustar os objetivos do projeto.

CONTROLE DO PROJETO

- Principais técnicas:

- Reuniões de acompanhamento;
- Gerenciamento das mudanças;
- EVA (*(Earned Value Analysis)*) ou EVM (*Earned Value Management*);
- Relatórios de acompanhamento;
- Controle de indicadores (CPI).

CONTROLAR O ESCOPO DO PROJETO

- Mudanças no escopo sempre irão ocorrer.
- Devemos nos aproximar aos fatores que criam mudanças nos projetos.
- Controlar o impacto das mudanças, integrando com outros processos de controle.

CONTROLAR CRONOGRAMA DO PROJETO

- Visa monitorar andamento do projeto para atualização do seu progresso e gerenciamento das mudanças feitas na linha de base do cronograma.
 - determinação da situação atual do cronograma do projeto;
 - influência nos fatores que criam mudanças no cronograma;
 - determinação de que o cronograma do projeto mudou;
 - gerenciamento das mudanças reais conforme ocorrem;
 - Calcular a previsibilidade do cronograma (Tendências).

CONTROLAR CRONOGRAMA DO PROJETO

○ Ferramentas e técnicas:

- Análise de cenários E-SE;
- Análise de desempenho;
- Análise de variação;
- Nivelamento de recursos;
- Compressão de cronograma e paralelismo das atividades.

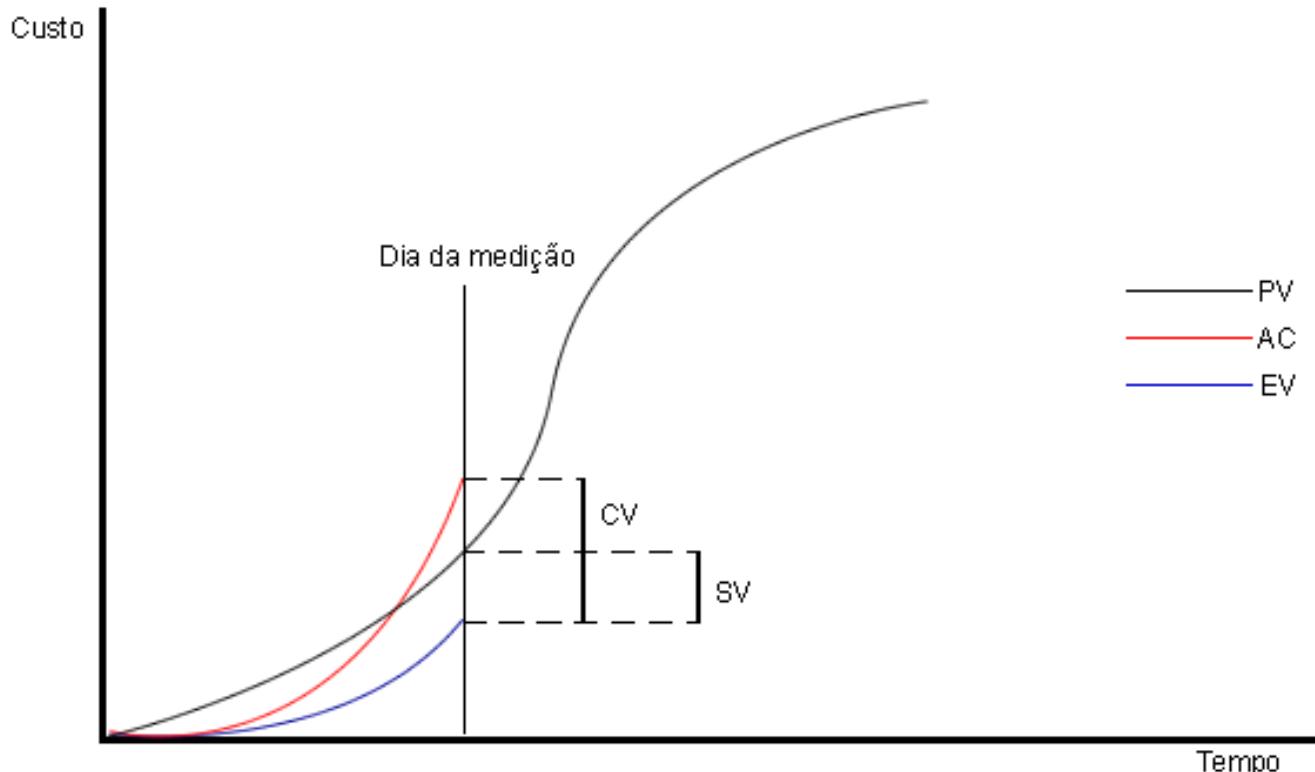
CONTROLAR CUSTO DO PROJETO

- Busca monitorar e controlar o desempenho financeiro do projeto, para que os custos fiquem dentro dos limites aceitáveis.
- Evitar que mudanças não aprovadas ou inadequadas sejam incluídas nos custos do projeto.
- Informar/aprovar partes interessadas.

CONTROLE DO PROJETO

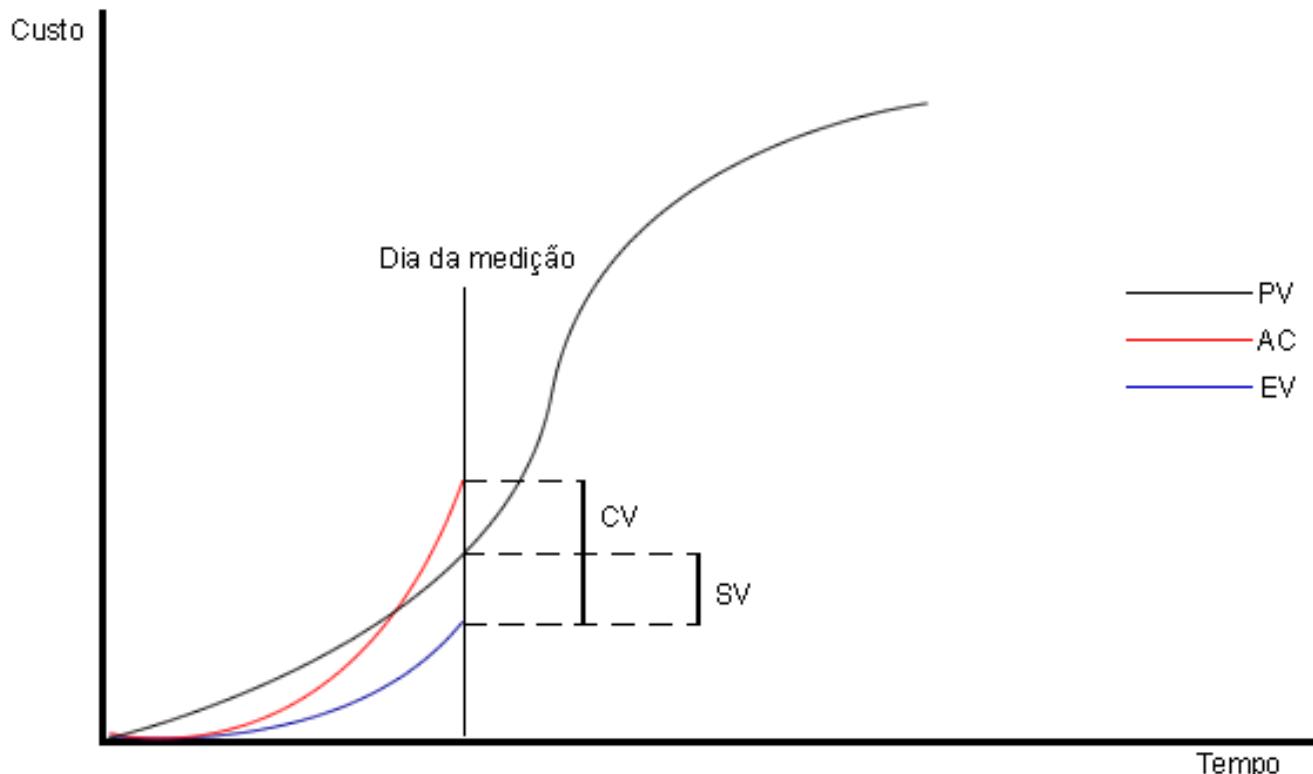
- **Valor planejado** (PV – Planned Value) – Custo planejado do projeto, relacionado à linha de base de custo do projeto.
- **Valor agregado** (EV – Earned Value) – Trata-se do custo planejado do projeto para o trabalho executado até o momento. Ou seja, é o valor dos serviços realmente executados até o momento.
- **Custo real** (AC – Actual Cost) – Custo total do trabalho até o momento.
- **Variação de prazo** (SV – Schedule Variance) – É a diferença entre o valor agregado até o momento (EV) e o valor planejado (PV).
 - $SV = EV - PV$
- **Variação de custo** (CV – Cost Variance) – É a diferença entre o valor agregado (EV) e o custo real (AC).
 - $CV = EV - AC$

CONTROLE DO PROJETO



- **CPI** (Cost Performance Index) – Trata-se do valor agregado do projeto(EV) dividido pelo custo real(AC) do mesmo, ambos até o momento do cálculo.
 - $CPI = EV / AC$

CONTROLE DO PROJETO



- SPI (Schedule Performance Index) – Trata-se do valor agregado até o momento dividido pelo valor planejado até o momento.
 - $\text{SPI} = \text{EV} / \text{PV}$

CONTROLE DO PROJETO - RELATÓRIO DE DESEMPENHO

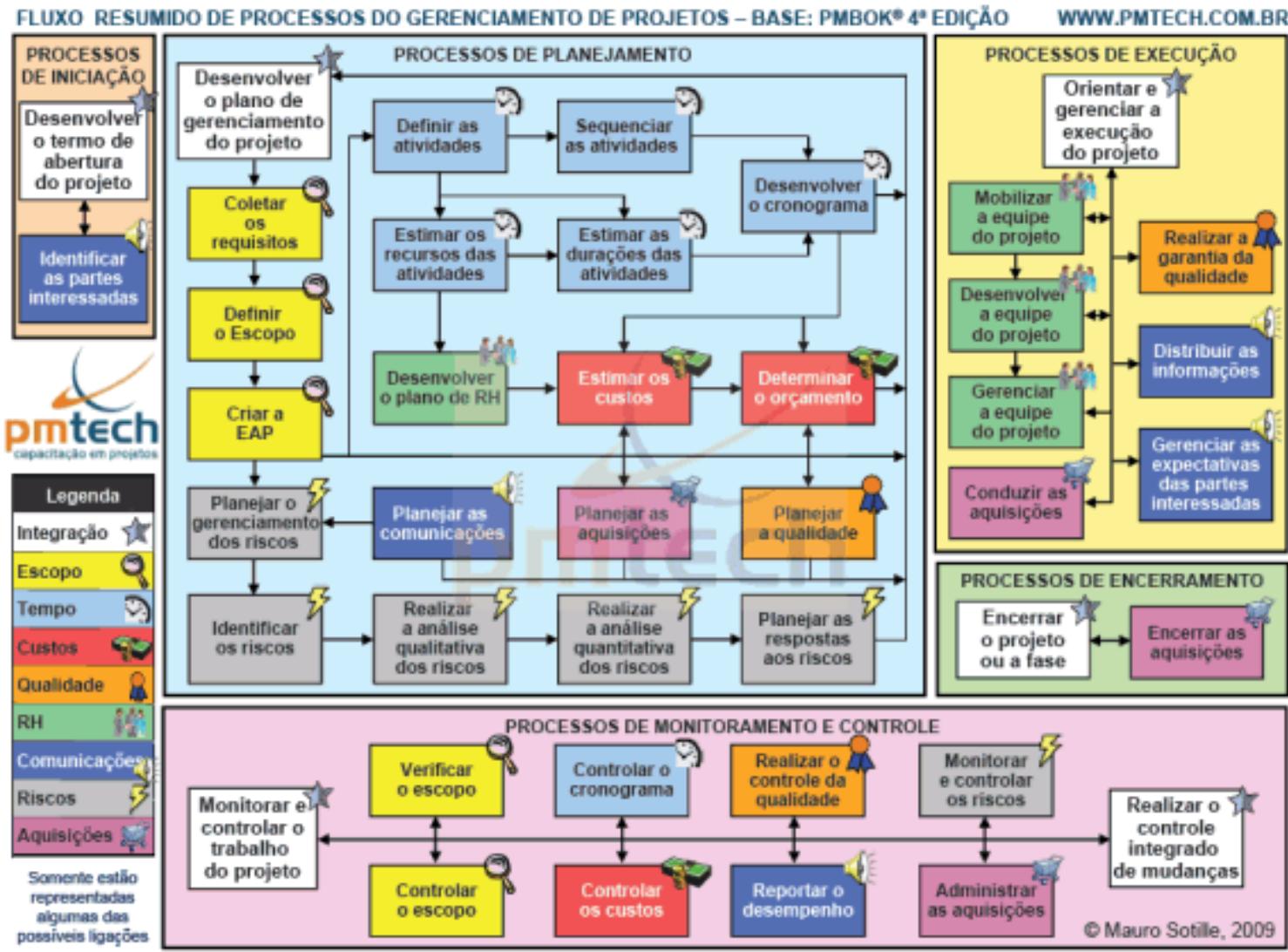
- Trata-se do relato da situação do projeto em um determinado período.
- Deve abordar o desempenho do projeto, relacionando o previsto x realizado x tendências.
- Aborda todas as áreas, principalmente:
 - Custo;
 - Prazo;
 - Escopo;
 - Riscos e;
 - Qualidade.

CONTROLE DO PROJETO

- Dependência com a estrutura da organização.

Características do projeto	Estrutura da organização	Funcional	Matricial			Por projeto
			Fraca	Balanceada	Forte	
Autoridade do gerente de projetos	Pouca ou nenhuma	Limitada	Baixa a moderada	Moderada a alta	Alta a quase total	
Disponibilidade de recursos	Pouca ou nenhuma	Limitada	Baixa a moderada	Moderada a alta	Alta a quase total	
Quem controla o orçamento do projeto	Gerente funcional	Gerente funcional	Misto	Gerente de projetos	Gerente de projetos	
Função do gerente de projetos	Tempo parcial	Tempo parcial	Tempo integral	Tempo integral	Tempo integral	
Equipe administrativa do gerenciamento de projetos	Tempo parcial	Tempo parcial	Tempo parcial	Tempo integral	Tempo integral	

ÁREAS DE CONHECIMENTO X GRUPOS DE PROCESSOS



Criado por Paulo Henrique Ladeira

EXECUÇÃO DO PROJETO

- Mobilizar a equipe do projeto
- Desenvolver a equipe do projeto
- Gerenciar a equipe do projeto
- Conduzir as aquisições
- Realizar a garantia da qualidade
- Distribuir as informações
- Gerenciar as expectativas das partes interessadas

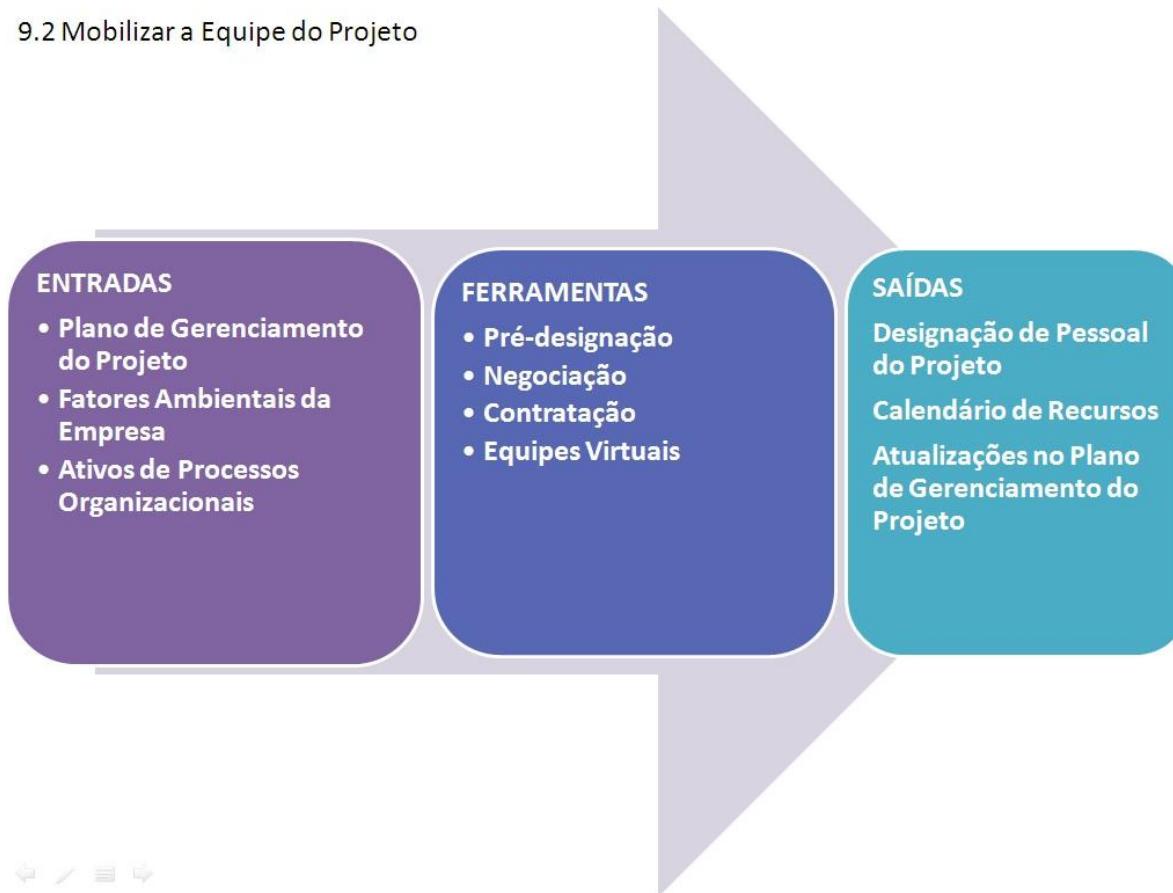


Criado por Paulo Henrique Ladeira

EXECUÇÃO DO PROJETO

○ Mobilizar a equipe do projeto

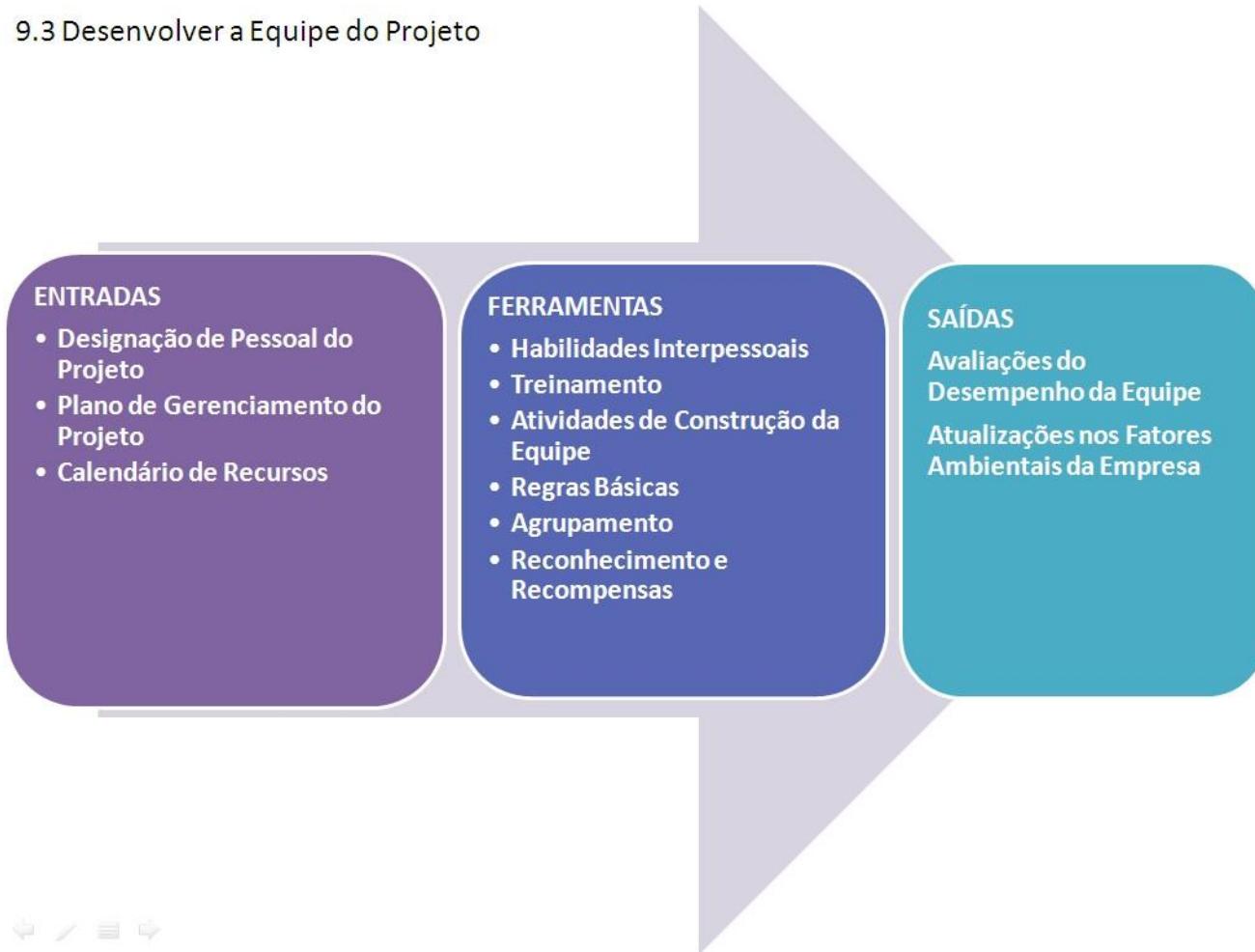
9.2 Mobilizar a Equipe do Projeto



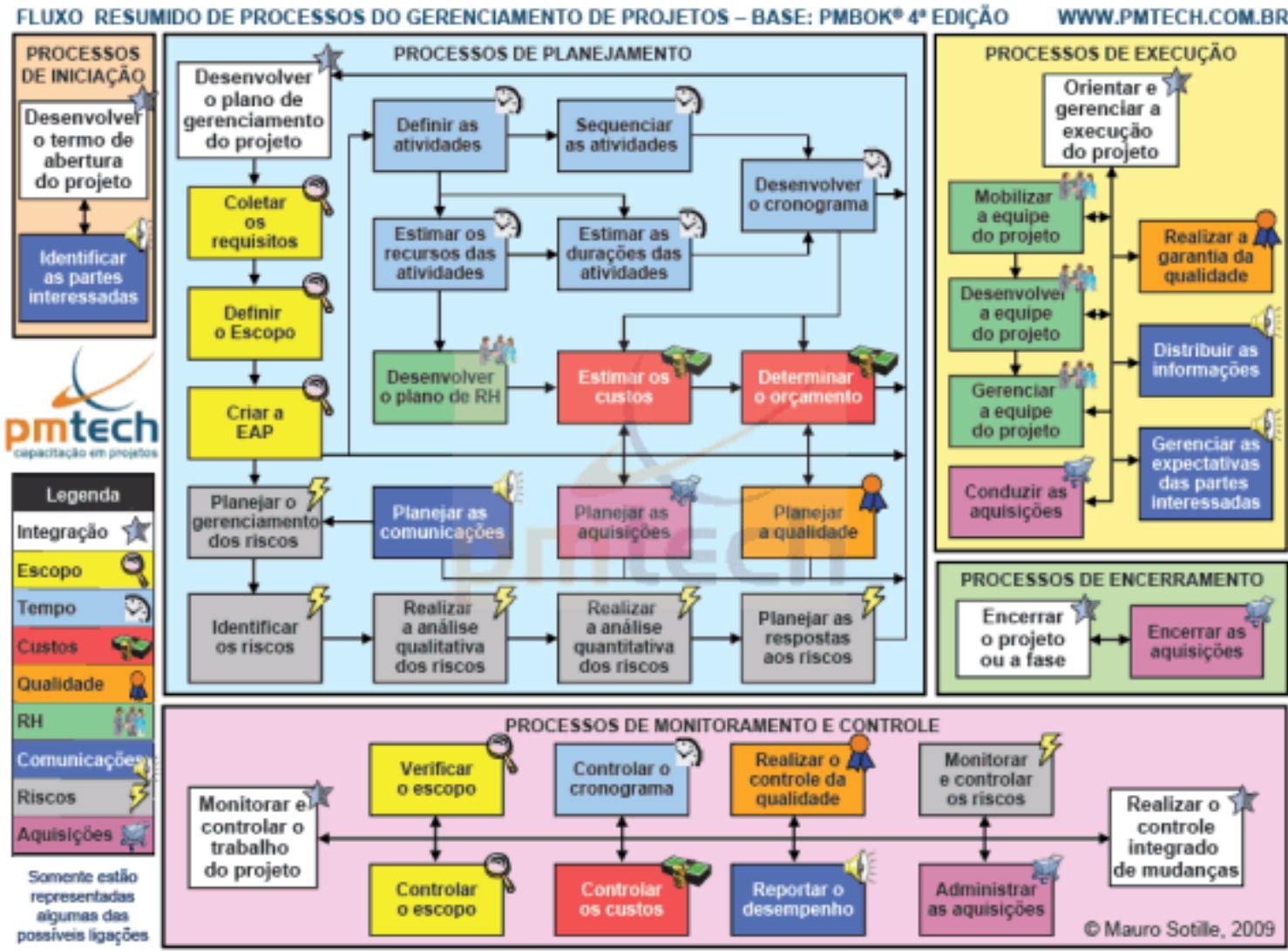
EXECUÇÃO DO PROJETO

○ Desenvolver a equipe do projeto

9.3 Desenvolver a Equipe do Projeto



ÁREAS DE CONHECIMENTO X GRUPOS DE PROCESSOS



Criado por Paulo Henrique Ladeira



ENCERRAMENTO DO PROJETO

- Consiste em concluir os trabalhos, dado que os resultados do projeto foram atingidos.
- Consiste em:
 - Entregar o resultado do projeto;
 - Entregar toda a documentação envolvida (interno e externo);
 - Recolher os aceites formais do projeto;
 - Pesquisa de satisfação do projeto;
 - Revisão do desempenho da equipe e atualização da base histórica;

ENCERRAMENTO DO PROJETO

Ocorre várias vezes dentro do projeto

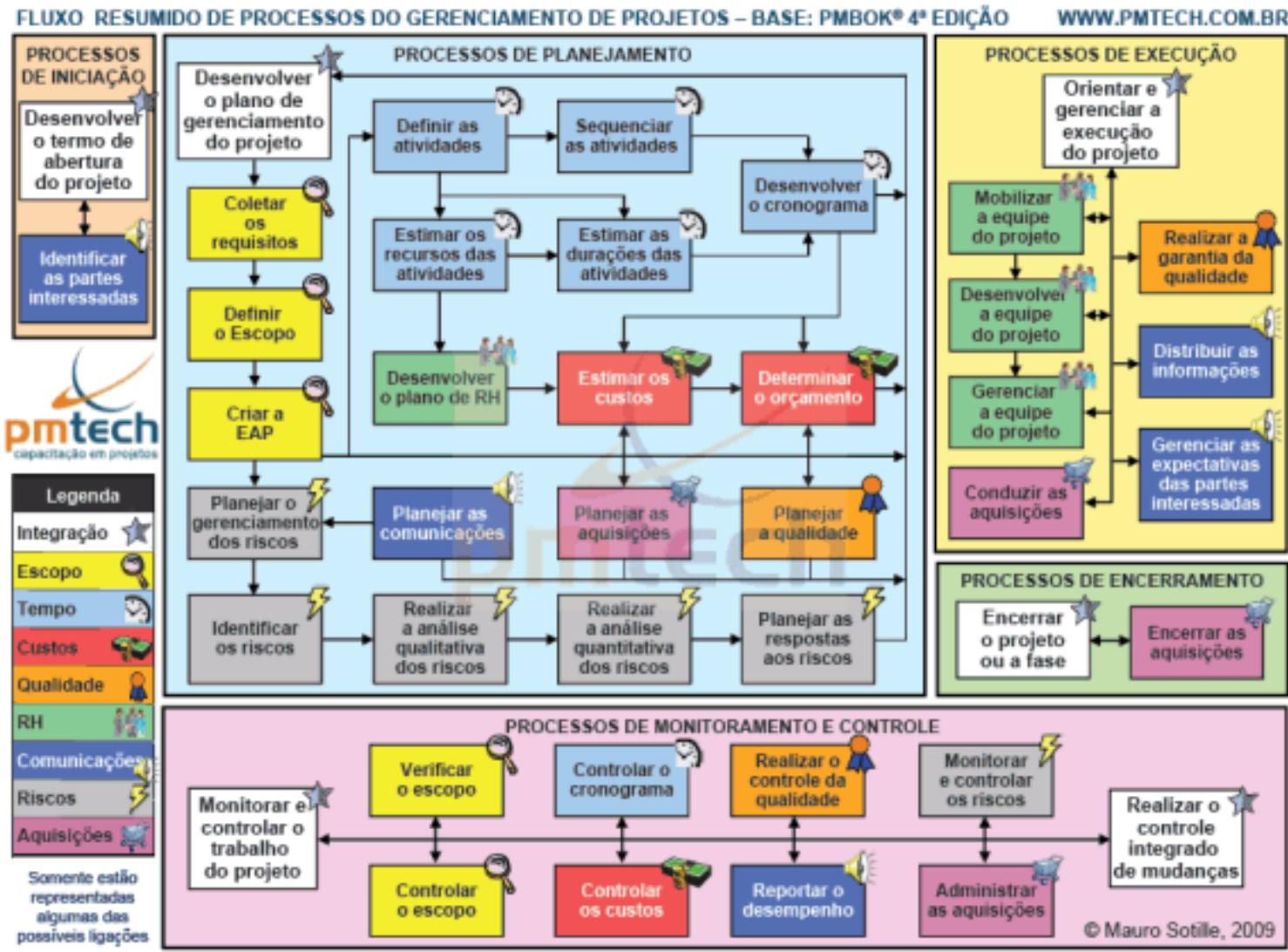
Implementação
+ Aceitação

Encerramento
Técnico

Encerramento
contratual

Encerramento
Administrativo

ÁREAS DE CONHECIMENTO X GRUPOS DE PROCESSOS

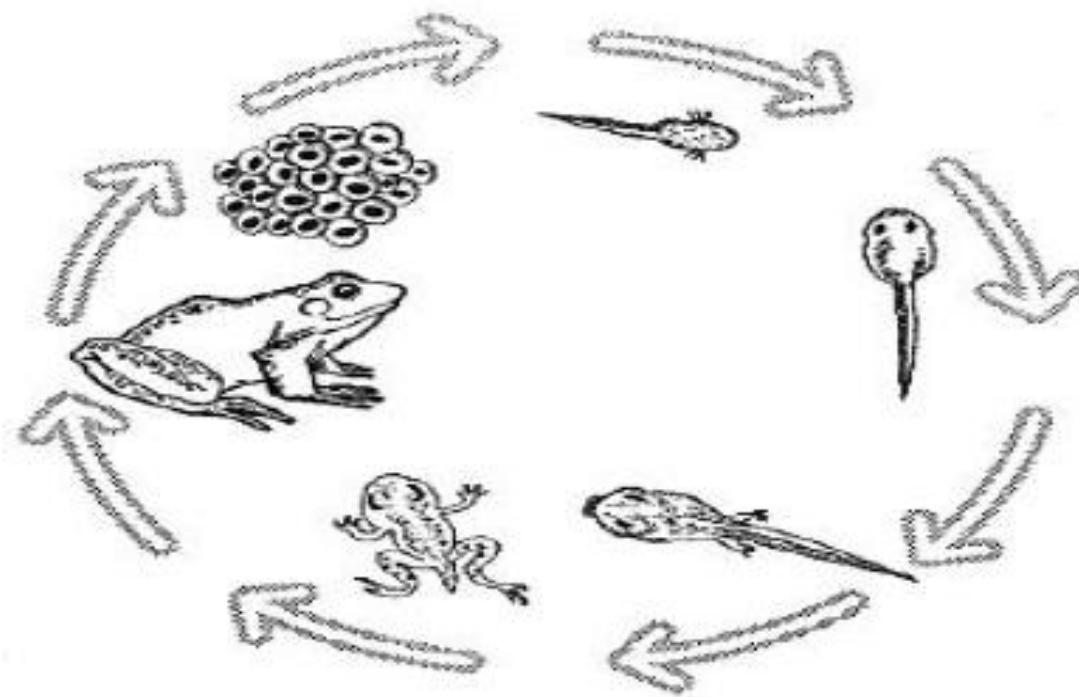


Criado por Paulo Henrique Ladeira

O CICLO DE VIDA

- O ciclo de vida define as fases que conectam o início de um projeto a seu fim.
- A organização em fases é um mecanismo que permite um maior controle gerencial.
- A transição de uma fase para outra normalmente é marcada por alguma entrega e/ou revisão.

O CICLO DE VIDA

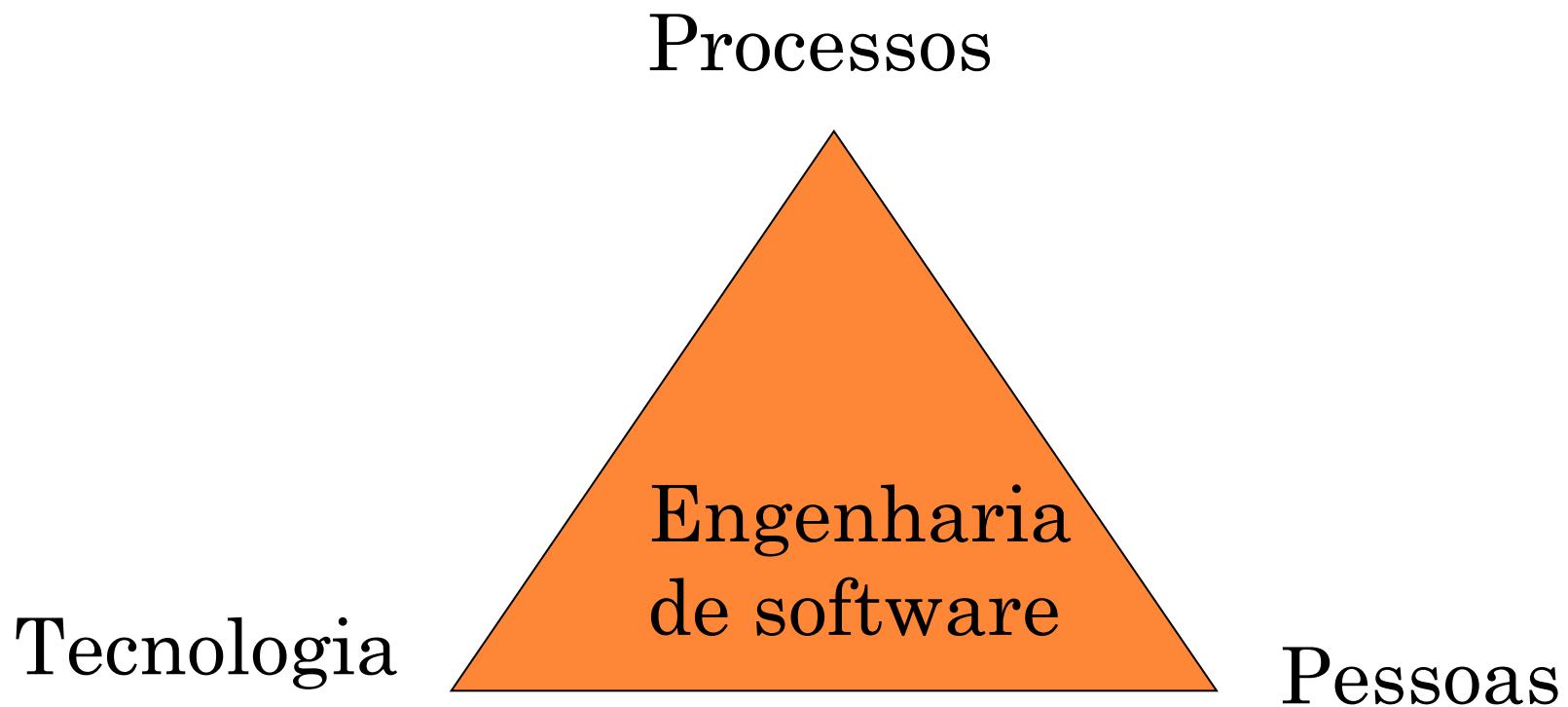


Criado por Paulo Henrique Ladeira

MODELO DE CICLO DE VIDA

- Um modelos de ciclo de vida serve como uma definição de alto-nível das fases que ocorrem durante o desenvolvimento.
- Não tem como objetivo prover informações detalhadas, mas somente destacar a atividades-chave e suas interdependências.

OS TRÊS PILARES DA ENGENHARIA DE SOFTWARE



PROCESSOS

- Conjunto de passos parcialmente ordenados:
 - Atividades, métodos, práticas e transformações;
 - Usados para atingir uma meta;
- Meta geralmente associada a um ou mais resultados concretos finais:
 - Produtos da execução do processo

PROCESSOS

- Deve possuir documentação que detalha:
 - O que é feito (produto)
 - Quando (passos)
 - Por quem (agentes)
 - O que usa (insumos)
 - O que produz (resultados)
- Um processo é uma receita seguida em um projeto, que concretiza uma abstração.

O QUE É PROCESSO DE SOFTWARE?

- É um conjunto de atividades cuja meta é o desenvolvimento ou evolução de software.
- Quais as principais atividades de processos composto em todo software?
 - Especificação – o que o sistema deve fazer e suas restrições de desenvolvimento.
 - Desenvolvimento – produção do sistema de software.
 - Validação – verificação de que o software é o que o cliente deseja.
 - Evolução – mudança do software em resposta às demandas de mudança.

O QUE É PROCESSO DE SOFTWARE?

- Deve ser documentado;
- Deve conter subdivisões que permitam avaliar o progresso e corrigir rumos
 - As subdivisões terminam em **marcos**:
 - São estados significativos do projeto
 - Permitem o acompanhamento preciso do projeto
- Contém marcos associados a **resultados concretos**, que são verificáveis
- O **produto** é um resultado associado ao marco de conclusão do projeto

O QUE É UM MODELO DE PROCESSO DE SOFTWARE?

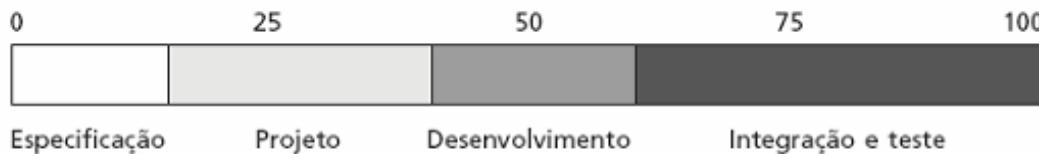
- Uma representação simplificada de um processo de software, apresentado sob uma perspectiva específica.
- Exemplos de modelos de processo são:
 - Modelo de *workflow* – seqüência de atividades;
 - Modelo de fluxo de dados – fluxo de informações;
 - Modelo de papel/ação – quem faz o quê.
- Modelos gerais de processo:
 - Cascata;
 - Desenvolvimento iterativo;
 - Engenharia de software baseada em componentes.

QUAIS SÃO OS CUSTOS DA ENGENHARIA DE SOFTWARE?

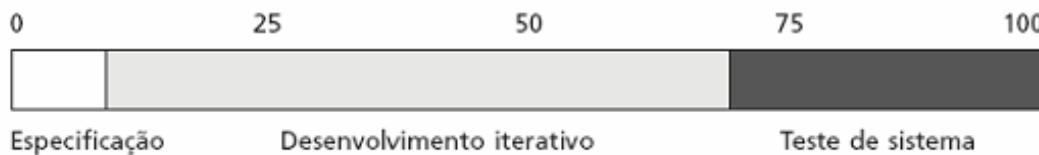
- Aproximadamente **60%** dos custos são custos de **desenvolvimento** e **40%** são custos de **testes**.
- Para software sob encomenda, os custos de evolução normalmente excedem de desenvolvimento.
- Outros fatores que afetam o custo:
 - Tipo de sistema que está sendo desenvolvido;
 - Os requisitos de atributos de sistema, tais como desempenho e confiabilidade;
 - O modelo de desenvolvimento que é usado.

DISTRIBUIÇÃO DE CUSTOS NAS ATIVIDADES

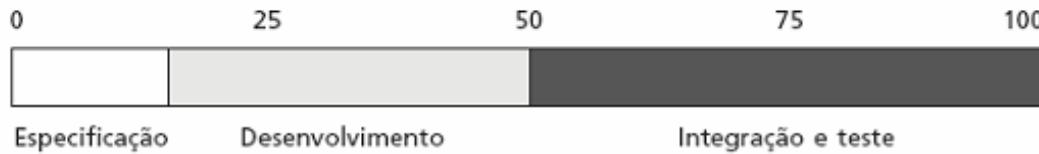
Modelo cascata



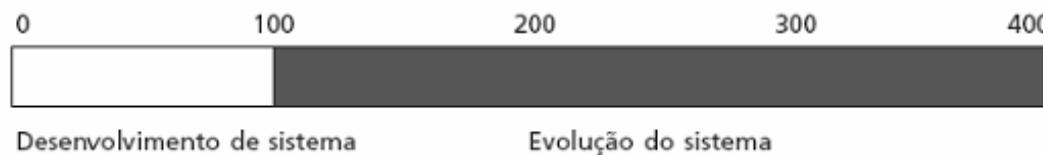
Desenvolvimento iterativo



Engenharia de software baseada em componentes



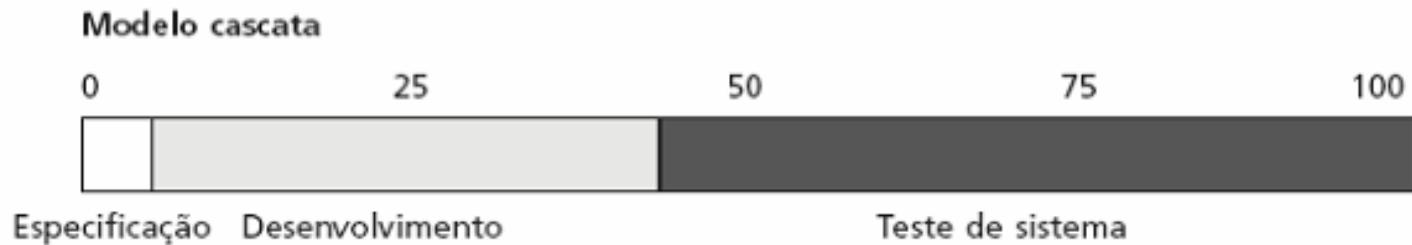
Custos de desenvolvimento e evolução ao longo da vida do software



Distribuição de custos nas atividades de engenharia de software

Fonte: Engenharia de software, 8th edição, Sommerville

CUSTOS DE DESENVOLVIMENTO DE PRODUTO



OBSTÁCULOS PARA UTILIZAÇÃO DE PROCESSOS

- Pensa-se que gera burocracia
- Pensa-se que o consumo de tempo é exagerado
- Não há entendimento da utilidade
- ...

SINTOMAS DA IMATURIDADE EM PROCESSOS

- Os projetos não são definidos com clareza e atividades de desenvolvimento são disfarçadas de manutenção (ambiente mais caótico).
- As pessoas não recebem o treinamento necessário ou escolhem arbitrariamente.
- As ferramentas não ajudam realmente a resolver os problemas.
- Os procedimentos e padrões, quando existem, são definidos e seguidos de forma “burocrática”.

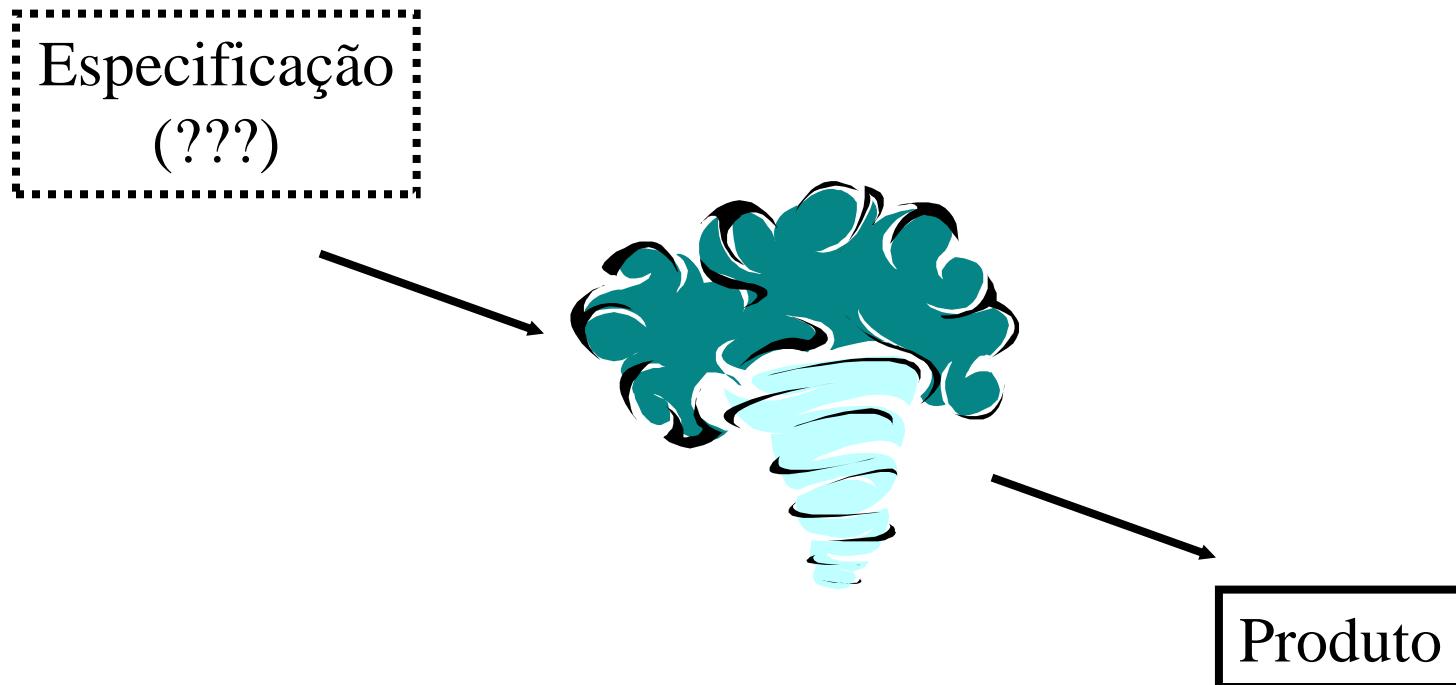
FORÇAS CAÓTICAS

- Ferramentas e técnicas vistos como magia ou bala de prata para resolução dos problemas.
- Existem gurus dentro da organização que dominam conteúdos intransferíveis.
- Os heróis utilizam processos informais que não podem ser transferidos.
- Na prática, esses heróis e gurus, salvam a organização com esforço pessoal, para suprir essas deficiências.

BOAS PRÁTICAS PARA PROCESSOS COM QUALIDADE

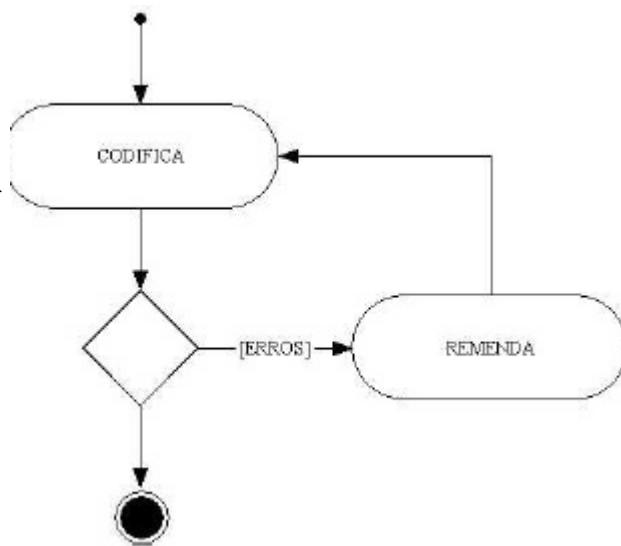
- Formação de grupos cuja atividade-fim é a melhoria dos processos.
- Formção de estruturas organizacionais adequadas e estáveis.
- Desenvolvimento de recursos de apoio a processos (padrões, modelos, exemplos, documentos de referência, base de dados, etc.).
- Realização de programas de treinamento, orientação, etc.
- Estabelecimento de medições do grau de progresso.
- Estabelecimento de mecanismos de controle e verificação.

CODIFICA-REMENDA



CODIFICA-REMENDA

Especificação
(**???**)



Produto

CODIFICA-REMENDA

- Não é uma metodologia
- Normalmente utilizado em projetos pequenos ou internos
- Não exige sofisticação técnica ou gerencial
 - Velocidade no primeiro momento
- Começa com uma idéia
 - Utiliza qualquer combinação de desenho, codificação, teste.
- Normalmente não é recomendado

CODIFICA-REMENDA

- Resulta em códigos péssimos
 - Utilizado se pretende-se fazer algum teste e descartar o código.
- Difícil de testar
- Difícil de utilizar se a empresa pretende produzir software de qualidade.
- Alto risco;
- Impossível de gerenciar;
- Não permite assumir compromissos confiáveis.

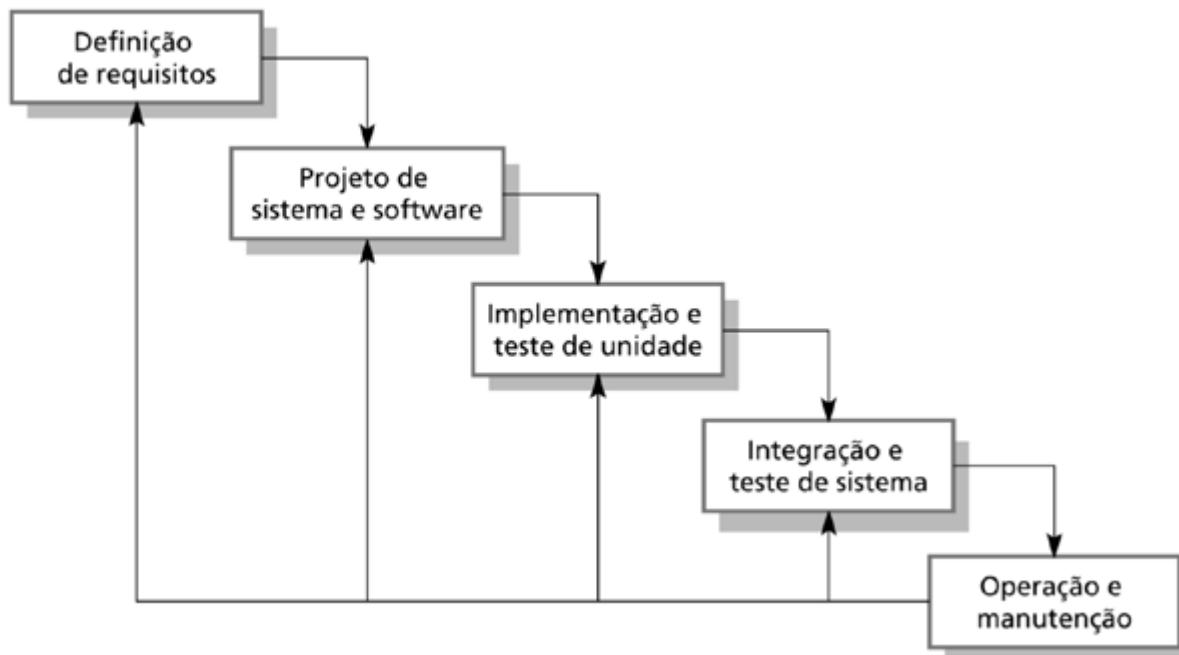
MODELOS GENÉRICOS DE PROCESSO DE SOFTWARE

- O modelo cascata
 - Fases separadas e distintas de especificação e desenvolvimento.
- Desenvolvimento evolucionário ou Modelos iterativos e incrementais
 - Especificação, desenvolvimento e validação são intercalados.
- Engenharia de software baseada em componentes
 - O sistema é montado a partir de componentes existentes.
- Existem muitas variantes destes modelos, por exemplo, o desenvolvimento formal onde um processo semelhante ao cascata é usado, mas a especificação é uma especificação formal, que é refinada durante os vários estágios para um projeto implementável.

MODELO DE CASCATA

Figura 4.1

Ciclo de vida de software.



MODELO CASCATA

- É talvez o modelo de desenvolvimento mais antigo.
- Chamado cascata porque cada atividade é feita em uma ordem pré-determinada
- Ao fim de cada fase é realizada uma revisão
- Enfatiza a importância da documentação
 - Requisitos e desenhos bem feitos
 - Todas as fases seguintes são dependentes desses documentos
- Similar ao processo para projetar e construir uma casa.

MODELO CASCATA

- Subprocessos executados em estrita seqüência:
 - pontos de controle bem definidos facilitam gestão;
- Teoricamente, confiável e utilizável em projetos de qualquer escala;
- Enfatiza a correção e a estabilidade dos requisitos logo no começo.
- Defensores acreditam que tempo gasto nas fases iniciais é recuperado com a redução de retrabalho.

MODELO CASCATA

- Interpretado literalmente, é rígido e burocrático;
- Requisitos, análise e desenho têm de ser muito bem dominados:
 - não são permitidos erros;
- Baixa visibilidade para o cliente:
 - só recebe o resultado final do projeto.

FASES DO MODELO CASCATA

- Análise e definição de requisitos
- Projeto de sistema e software
- Implementação e teste de unidade
- Integração e teste de sistema
- Operação e manutenção

VANTAGENS DO MODELO DE CASCATA

- Útil quando ocorre mudanças na equipe
- Bom para o aprendizado de pessoas inexperientes da equipe.

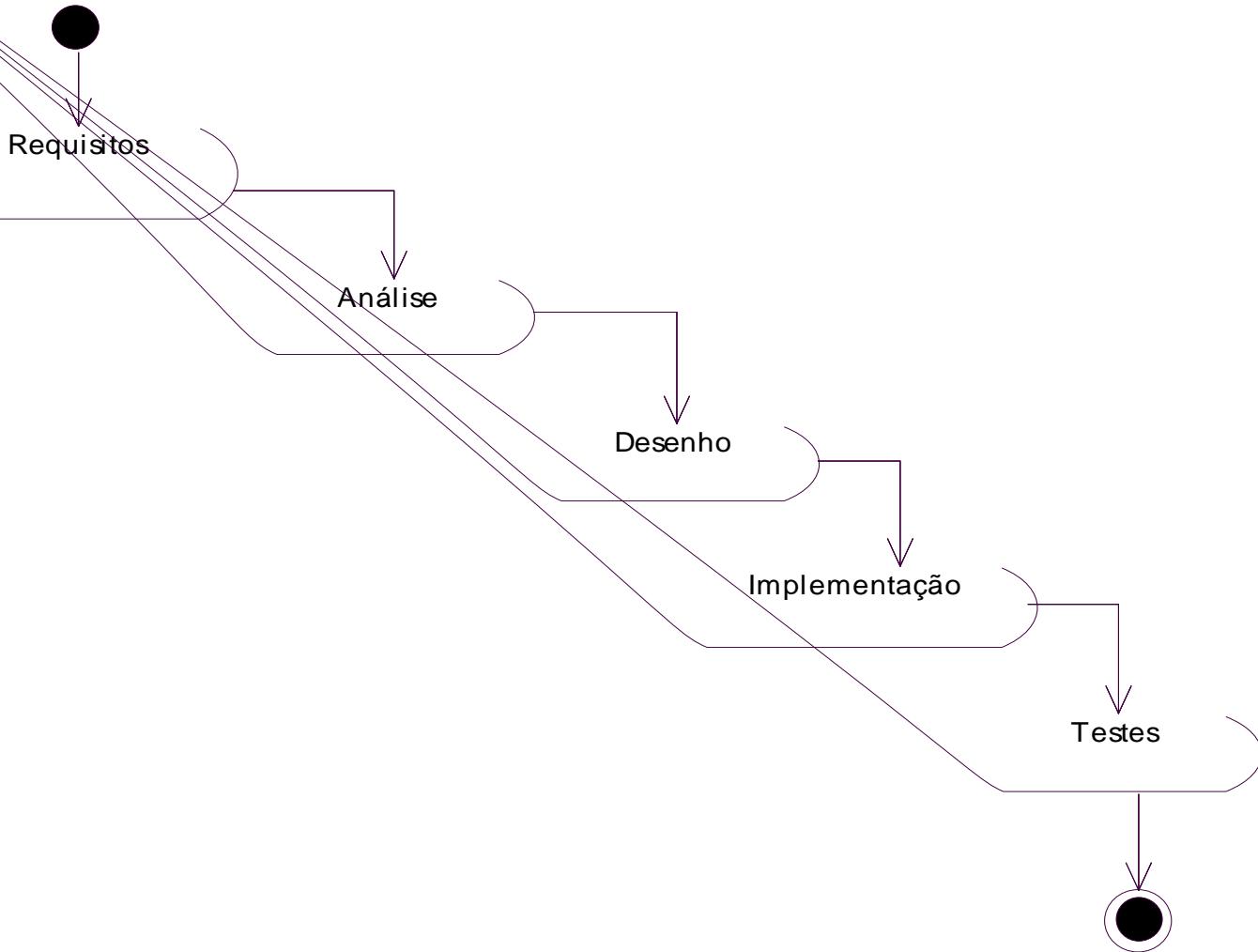
DESVANTAGENS DO MODELO DE CASCATA

- Particionamento inflexível do projeto em estágios distintos, dificulta a resposta aos requisitos de mudança do cliente.
- Portanto, este modelo é apropriado somente quando os requisitos são bem compreendidos, e quando as mudanças forem bastante limitadas durante o desenvolvimento do sistema.
- Poucos sistemas de negócio têm requisitos estáveis.
- O modelo cascata é o mais usado em projetos de engenharia de sistemas de grande porte, onde um sistema é desenvolvido em várias localidades.
- Defeitos encontrados tardeamente.

CASCATA

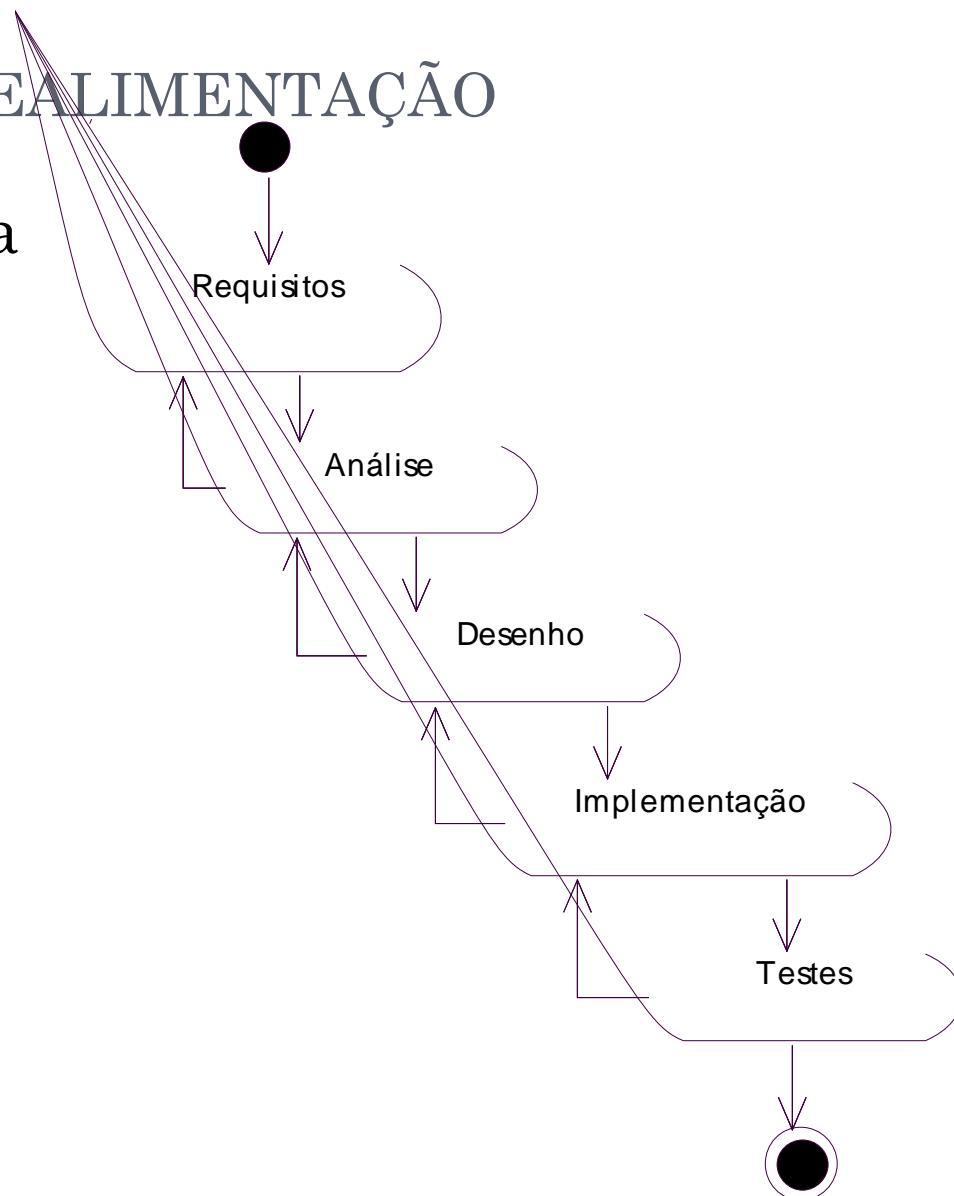
- Existem inúmeras variações dos modelos em cascata
- Outras desvantagens:
 - Assume que todos terão o mesmo entendimento da documentação
 - Difícil que os requisitos estejam corretos e não se modifiquem
 - Após utilizar o sistema usuários irão ter novas idéias.
 - Defeitos encontrados tardeamente
- Vantagens
 - Útil quando ocorre mudanças na equipe
 - Bom para o aprendizado de pessoas inexperientes da equipe.

CASCATA

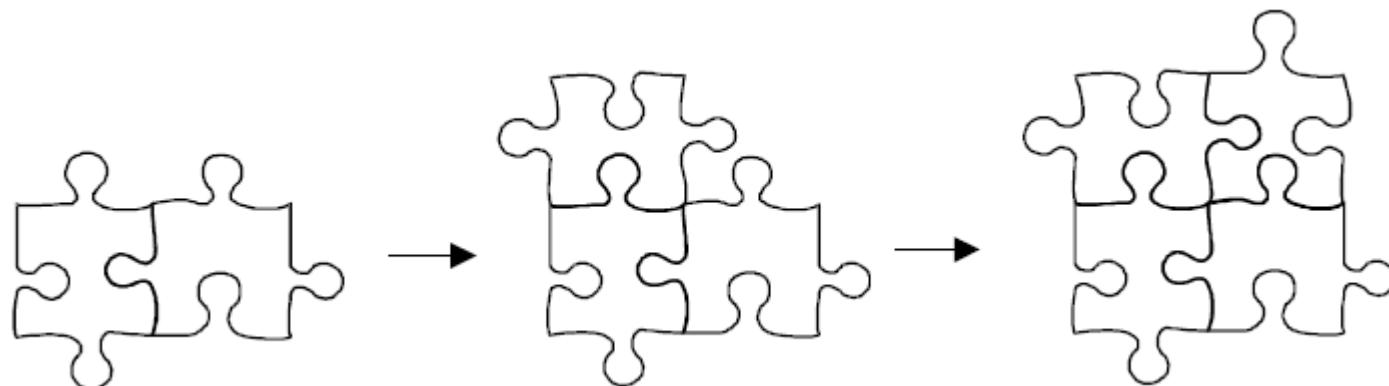


CASCATA COM REALIMENTAÇÃO

- Realimentação a cada etapa
- Mais realista
- Dificulta o gerenciamento



MODELOS ITERATIVOS E INCREMENTAIS



MODELOS ITERATIVOS E INCREMENTAIS

- Criado para resolver os problemas dos modelos em cascata.
- Cada iteração continua sobre a iteração seguinte.
- Encoraja feedback dos usuários através da utilização de protótipos.
- Para ser considerado iterativo precisa:
 - Cada iteração tem por base a anterior para refinar o produto final.
 - Cada iteração é construída sobre a anterior até que tudo esteja construído.

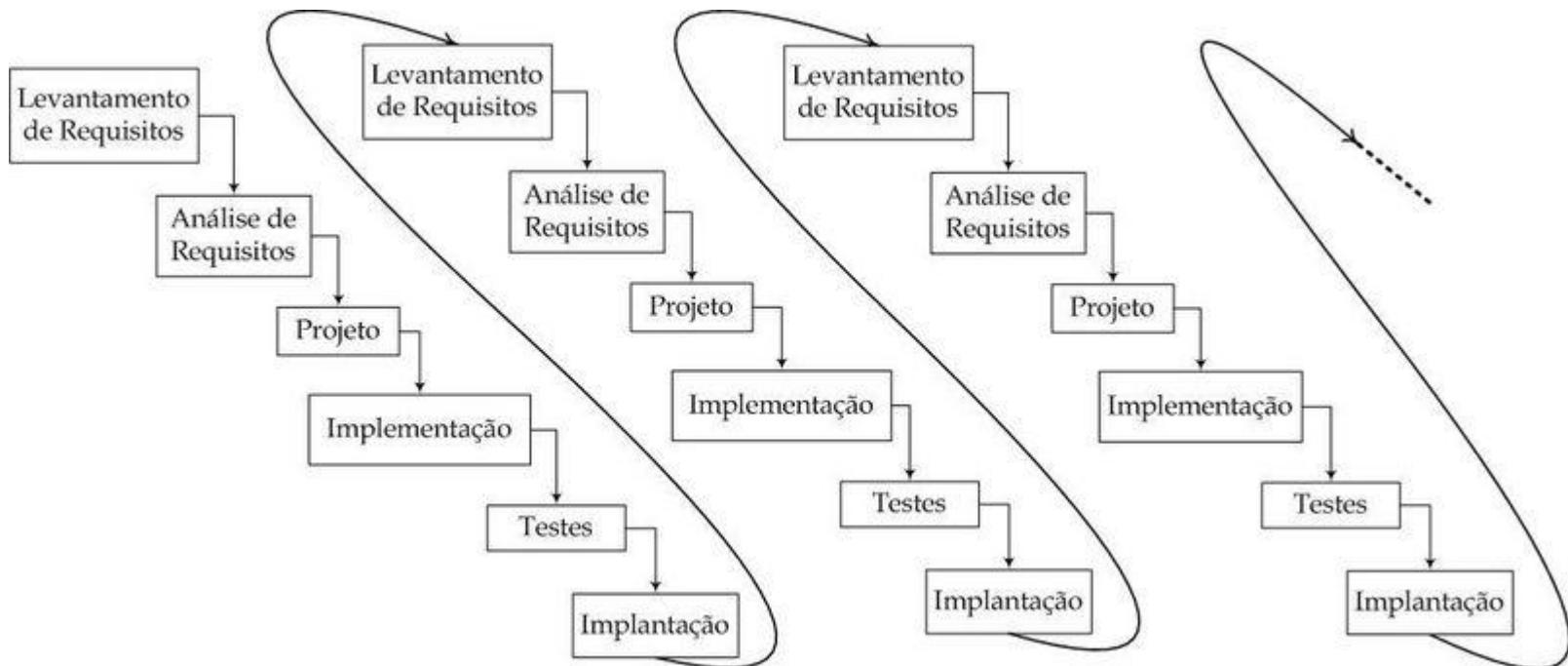


MODELOS ITERATIVOS E INCREMENTAIS

- Reconhece que modificações nos requisitos irão ocorrer.
- Provê oportunidades explícitas para rever:
 - Requisitos
 - Desenho
 - Implementação
 - Qualidade
- Utiliza protótipos para reduzir o risco.
- Defensores dos modelos iterativos acreditam que os processos têm que ser flexíveis.

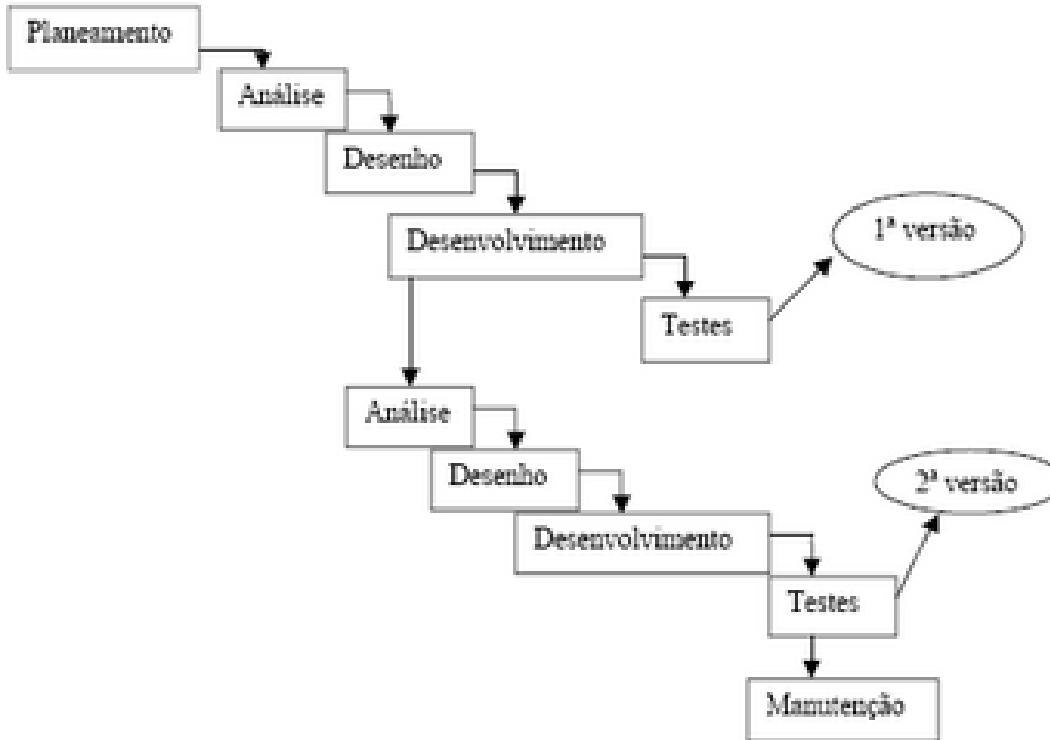


MODELOS ITERATIVOS E INCREMENTAIS



Criado por Paulo Henrique Ladeira

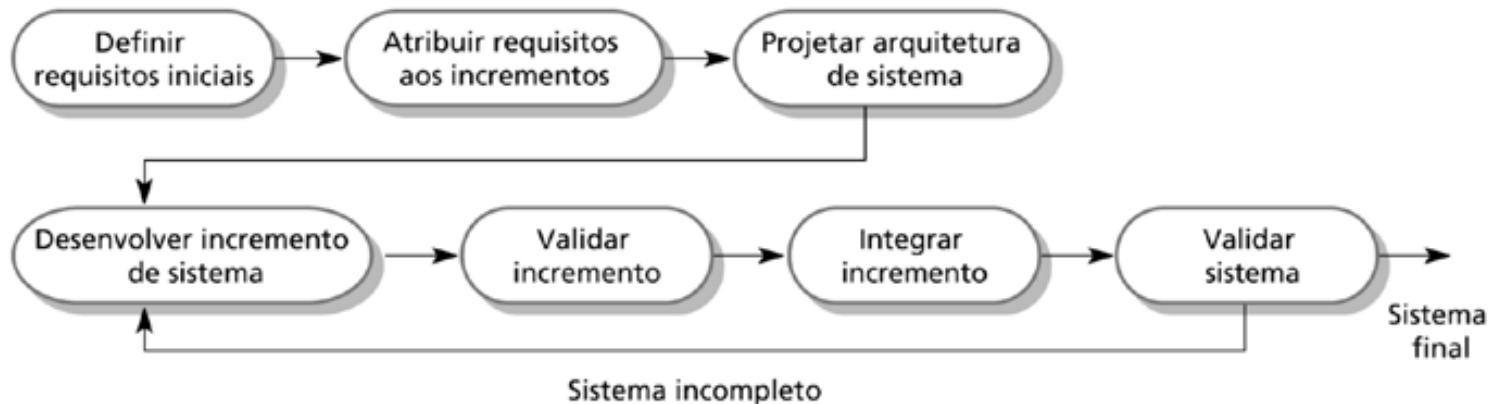
MODELOS ITERATIVOS E INCREMENTAIS



MODELOS ITERATIVOS E INCREMENTAIS

Figura 4.4

Entrega incremental.



MODELOS ITERATIVOS E INCREMENTAIS

○ Vantagens

- Permite rever os requisitos, desenho e codificação.
- O valor pode ser entregue para o cliente com cada incremento e, desse modo, a funcionalidade do sistema é disponibilizada mais cedo.
- O incremento inicial age como um protótipo para auxiliar a eliciar os requisitos para incrementos posteriores do sistema.
- Riscos menores de falha geral do projeto.
- Os serviços de sistema de mais alta prioridade tendem a receber mais testes.



MODELOS ITERATIVOS E INCREMENTAIS

○ Desvantagens

- Difícil de utilizar em projetos com custo fixo.
 - Requisitos se modificam
- Codificação inicia-se antes do requisito estar fechado:
 - Código pode ter que ser descartado.
- Falta de visibilidade de processo;
- Os sistemas são freqüentemente mal estruturados;
- Habilidades especiais (por exemplo, em linguagens para prototipação rápida) podem ser solicitadas.



MODELO ESPIRAL

- O processo é representado como uma espiral ao invés de uma seqüência de atividades com realimentação.
- Cada *loop* na espiral representa uma fase no processo.
- Sem fases definidas, tais como especificação ou projeto – os *loops* na espiral são escolhidos dependendo do que é requisitado.
- Os riscos são explicitamente avaliados e resolvidos ao longo do processo.

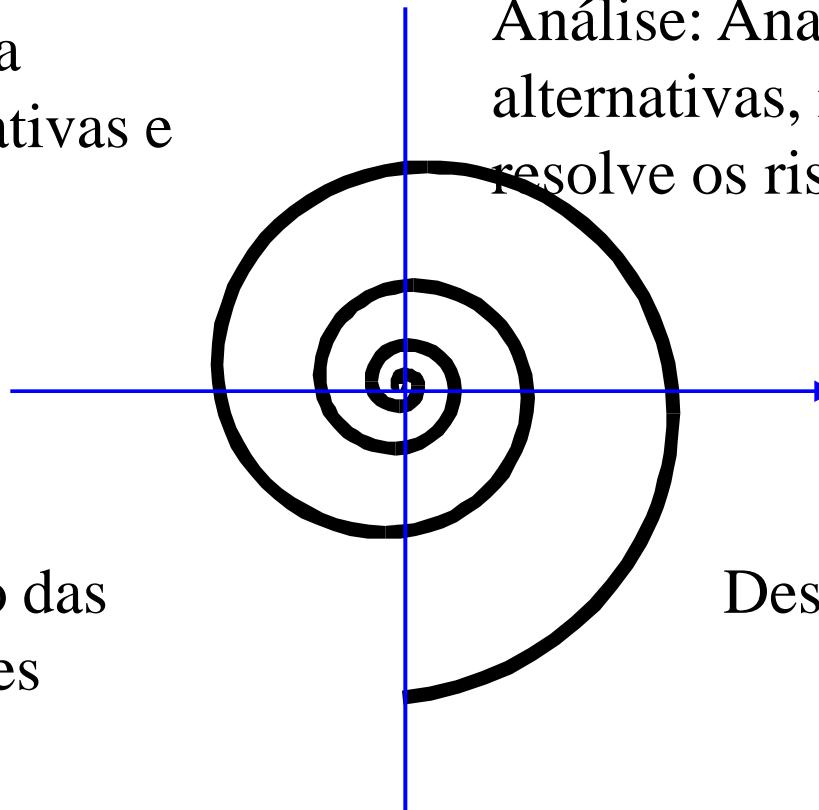
MODELO ESPIRAL

Início: Determina objetivos, alternativas e limitações

Análise: Analisa alternativas, identifica e resolve os riscos

Planejamento das próximas fases

Desenvolvimento



MODELO ESPIRAL

- Definição de objetivos
 - Objetivos específicos para a fase são identificados.
- Avaliação e redução de riscos
 - Riscos são avaliados e atividades são realizadas para reduzir os riscos-chave.
- Desenvolvimento e validação
 - Um modelo de desenvolvimento para o sistema, que pode ser qualquer um dos modelos genéricos, é escolhido.
- Planejamento
 - O projeto é revisado e a próxima fase da espiral é planejada.

DESENVOLVIMENTO RÁPIDO DE SOFTWARE



OBJETIVOS

- Explicar como um processo de desenvolvimento incremental e iterativo conduz a uma entrega mais rápida de software mais útil
- Discutir a essência dos métodos de desenvolvimento ágeis
- Explicar os princípios e as práticas da *extreme programming*
- Explicar os papéis da prototipação no processo de software



TÓPICOS ABORDADOS

- Métodos ágeis
- *Extreme programming*
- Desenvolvimento rápido de aplicações
- Prototipação de software



DESENVOLVIMENTO RÁPIDO DE SOFTWARE

- Devido à rápida mudança dos ambientes de negócio, os negócios devem responder às novas oportunidades e à competição.
- Isso requer software e desenvolvimento rápido, e a entrega é, freqüentemente, o requisito mais crítico para sistemas de software.
- Os negócios podem estar dispostos a aceitar um software de baixa qualidade se a entrega rápida e a funcionalidade essencial for possível.



REQUISITOS

- Devido às mudanças de ambiente, é praticamente impossível chegar a um conjunto de requisitos de sistema estáveis e consistentes.
- Portanto, um modelo de desenvolvimento em cascata não é prático, e uma abordagem para desenvolvimento baseada em especificação e entrega iterativas é a única alternativa para entregar software rapidamente.



CARACTERÍSTICAS DOS PROCESSOS RAD

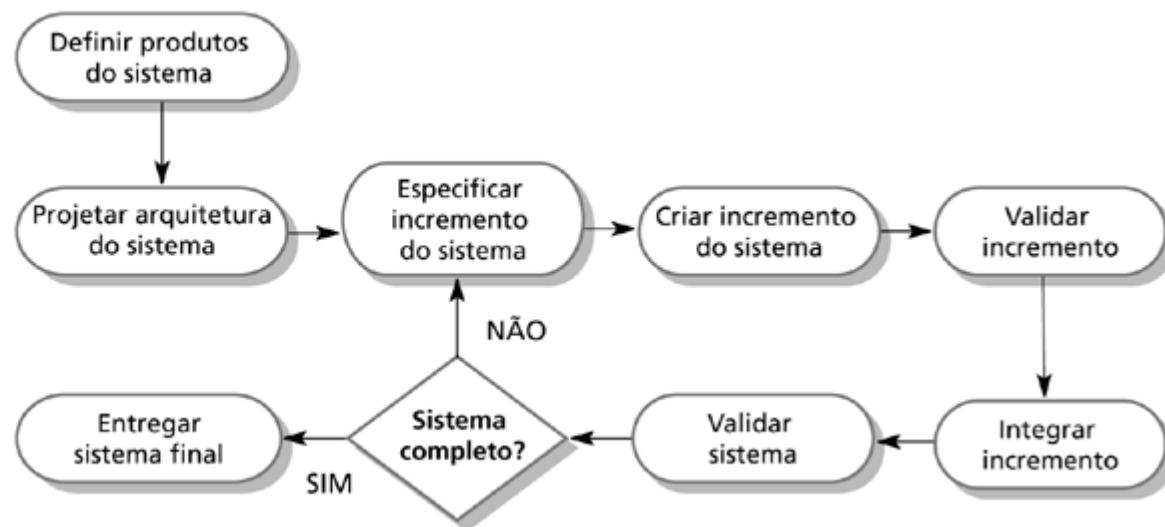
- Os processos de especificação, de projeto e de implementação são concorrentes; não há uma especificação detalhada e a documentação de projeto é minimizada.
- O sistema é desenvolvido em uma série de incrementos. Os usuários finais avaliam cada incremento e fazem propostas para incrementos posteriores.
- As interfaces de usuário de sistema são geralmente desenvolvidas usando-se um sistema de desenvolvimento interativo.



UM PROCESSO DE DESENVOLVIMENTO ITERATIVO

Figura 17.1

Processo de desenvolvimento iterativo.



VANTAGENS DO DESENVOLVIMENTO INCREMENTAL

- **Entrega acelerada dos serviços de cliente.** Cada incremento fornece a funcionalidade de mais alta prioridade para o cliente.
- **Engajamento do usuário com o sistema.**
Os usuários têm de estar envolvidos no processo de desenvolvimento, o que significa que o sistema muito provavelmente atenderá aos seus requisitos, e que os usuários estarão mais comprometidos com ele.



PROBLEMAS COM DESENVOLVIMENTO INCREMENTAL

○ Problemas de gerenciamento

- O progresso pode ser difícil de julgar e os problemas, difíceis de encontrar, porque não há documentação que mostre o que foi feito.

○ Problemas contratuais

- O contrato normal pode incluir uma especificação; sem uma especificação, formulários diferentes de contrato têm de ser usados.

○ Problemas de validação

- Sem uma especificação, contra o que o sistema está sendo testado?

○ Problemas de manutenção

- Mudanças contínuas tendem a corromper a estrutura do software, o que torna mais dispendioso mudar e evoluir para atender aos novos requisitos.



MÉTODOS ÁGEIS

- A insatisfação com os overheads envolvidos nos métodos de projeto levou à criação dos métodos ágeis. Esses métodos:
 - Enfocam o código ao invés do projeto;
 - São baseados na abordagem iterativa para desenvolvimento de software;
 - São destinados a entregar software de trabalho e evoluí-lo rapidamente para atender aos requisitos que se alteram.
- Os métodos ágeis são, provavelmente, os mais adequados para sistemas de negócio de porte pequeno/médio ou produtos para PC.



PRINCÍPIOS DOS MÉTODOS ÁGEIS

Tabela 17.1 Princípios dos métodos ágeis

Princípio	Descrição
Envolvimento do cliente	Clientes devem ser profundamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar as iterações do sistema.
Entrega incremental	O software é desenvolvido em incrementos e o cliente especifica os requisitos a serem incluídos em cada incremento.
Pessoas, não processo	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Os membros da equipe devem desenvolver suas próprias maneiras de trabalhar sem processos prescritivos.
Aceite as mudanças	Tenha em mente que os requisitos do sistema vão mudar, por isso projete o sistema para acomodar essas mudanças.
Mantenha a simplicidade	Concentre-se na simplicidade do software que está sendo desenvolvido e do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.



PROBLEMAS COM MÉTODOS ÁGEIS

- Pode ser difícil manter o interesse dos clientes que estão envolvidos no processo.
- Os membros da equipe podem ser inadequados para o intenso envolvimento que caracteriza os métodos ágeis.
- A priorização de mudanças pode ser difícil onde existem múltiplos stakeholders/sponsors.
- A manutenção da simplicidade requer trabalho extra.
- Do mesmo modo que nas outras abordagens para desenvolvimento iterativo, os contratos podem ser um problema.



EXTREME PROGRAMMING

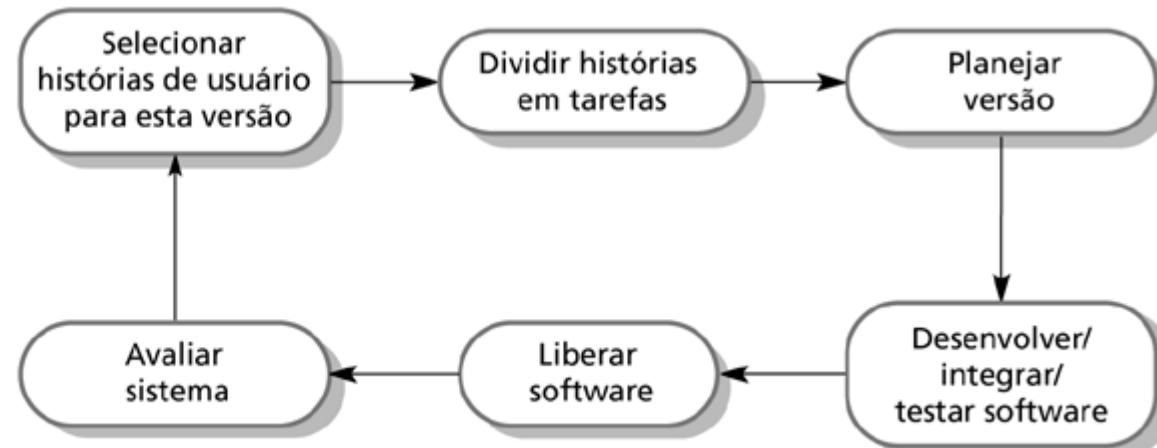
- É talvez o mais conhecido e mais amplamente usado dos métodos ágeis.
- A extreme programming (XP) leva uma abordagem ‘extrema’ para desenvolvimento iterativo.
 - Novas versões podem ser compiladas várias vezes por dia;
 - Os incrementos são entregues para os clientes a cada 2 semanas;
 - Todos os testes devem ser realizados para cada nova versão e a nova compilação somente é aceita se todos os testes forem realizados com sucesso.



O CICLO DE *RELEASE* EM XP

Figura 17.3

Ciclo de um release em *extreme programming*



PRÁTICAS DO *EXTREME PROGRAMMING*

Tabela 17.2 Práticas da *extreme programming*

Princípio ou prática	Descrição
Planejamento incremental	Os requisitos são registrados em cartões de histórias e as histórias a serem incluídas em um release são determinadas pelo tempo disponível e sua prioridade relativa. Os desenvolvedores dividem essas histórias em 'tarefas'. Veja as figuras 17.4 e 17.5.
Pequenos releases	O conjunto mínimo útil de funcionalidade que agrega valor ao negócio é desenvolvido primeiro. Releases do sistema são freqüentes e adicionam funcionalidade incrementalmente ao primeiro release.
Projeto simples	É realizado um projeto suficiente para atender aos requisitos atuais e nada mais.
Desenvolvimento <i>test-first</i>	Um framework automatizado de teste unitário é usado para escrever os testes para uma nova parte da funcionalidade antes que esta seja implementada.
Refactoring	Espera-se que todos os desenvolvedores recriem o código continuamente tão logo os aprimoramentos do código forem encontrados. Isso torna o código simples e fácil de manter.
Programação em pares	Os desenvolvedores trabalham em pares, um verificando o trabalho do outro e fornecendo apoio para realizar sempre um bom trabalho.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de tal maneira que não se formem ilhas de conhecimento, com todos os desenvolvedores de posse de todo o código. Qualquer um pode mudar qualquer coisa.
Integração contínua	Tão logo o trabalho em uma tarefa seja concluído, este é integrado ao sistema como um todo. Depois de qualquer integração, todos os testes unitários do sistema devem ser realizados.
Ritmo sustentável	Grandes quantidades de horas extras não são consideradas aceitáveis, pois, no médio prazo, há uma redução na qualidade do código e na produtividade.
Cliente on-site	Um representante do usuário final do sistema (o cliente) deve estar disponível em tempo integral para apoiar a equipe de XP. No processo da <i>extreme programming</i> , o cliente é um membro da equipe de desenvolvimento e é responsável por trazer os requisitos do sistema à equipe para implementação.

RATIONAL UNIFIED PROCESS (RUP)

- É um modelo de processo moderno derivado do trabalho sobre a UML e do Processo Unificado de Desenvolvimento de Software associado.
- Traz elementos de todos os modelos de processos genéricos visto.

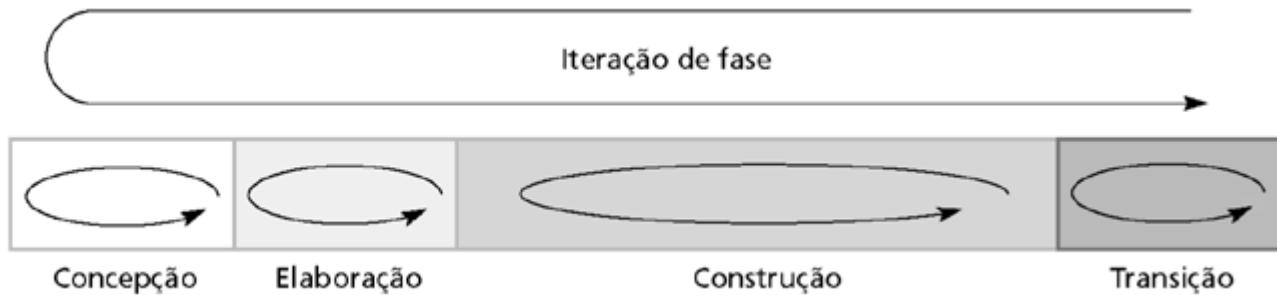
RATIONAL UNIFIED PROCESS (RUP)

- Normalmente descrito a partir de três perspectivas:
 - Uma perspectiva dinâmica que mostra as fases ao longo do tempo;
 - Uma perspectiva estática que mostra atividades de processo;
 - Uma perspectiva prática que sugere boas práticas.

RATIONAL UNIFIED PROCESS (RUP)

Figura 4.12

Fases no Rational Unified Process.



RATIONAL UNIFIED PROCESS (RUP)

○ Concepção

- Estabelecer o *business case* para o sistema.
- Identifica todas as entidades envolvidas (pessoas e sistemas).
- Análise de viabilidade do projeto.

○ Elaboração

- Desenvolver um entendimento do domínio do problema e a arquitetura do sistema.
- Desenvolve-se um plano do projeto e os maiores riscos são identificados.
- Uma possível saída dessa fase é um modelo de requisitos, com casos de uso, arquitetura, plano de desenvolvimento, etc.

RATIONAL UNIFIED PROCESS (RUP)

○ Construção

- Projeto, programação e teste de sistema.
- Ao término dessa fase deve existir um software funcionando e com toda a documentação associada.

○ Transição

- Implantar o sistema no seu ambiente operacional.
- Muitas vezes ignorada nos outros modelos, apesar de se tratar de uma atividade, muitas vezes, cara e problemática.

RATIONAL UNIFIED PROCESS (RUP)

- A visão estática prioriza atividades que ocorrem durante o processo de desenvolvimento, chamadas de workflow.
- É dividida em workflows centrais (6) e de apoio (3).

RATIONAL UNIFIED PROCESS (RUP)

Tabela 4.1 Workflows estáticos no Rational Unified Process.

Workflow	Descrição
Modelagem de negócios	Os processos de negócios são modelados usando casos de uso de negócios.
Requisitos	Os agentes que interagem com o sistema são identificados e os casos de uso são desenvolvidos para modelar os requisitos de sistema.
Análise e projeto	Um modelo de projeto é criado e documentado usando modelos de arquitetura, modelos de componente, modelos de objeto e modelos de seqüência.
Implementação	Os componentes de sistema são implementados e estruturados em subsistemas de implementação. A geração automática de código com base nos modelos de projeto ajuda a acelerar esse processo.
Teste	O teste é um processo iterativo realizado em conjunto com a implementação. O teste de sistema segue o término da implementação.
Implantação	Uma versão do produto é criada, distribuída aos usuários e instalada no local de trabalho.
Gerenciamento de configuração e mudanças	Este workflow de apoio gerencia as mudanças do sistema (veja o Capítulo 29).
Gerenciamento de projetos	Este workflow de apoio gerencia o desenvolvimento do sistema (veja o Capítulo 5).
Ambiente	Este workflow está relacionado à disponibilização de ferramentas apropriadas de software para a equipe de desenvolvimento.

RATIONAL UNIFIED PROCESS (RUP)

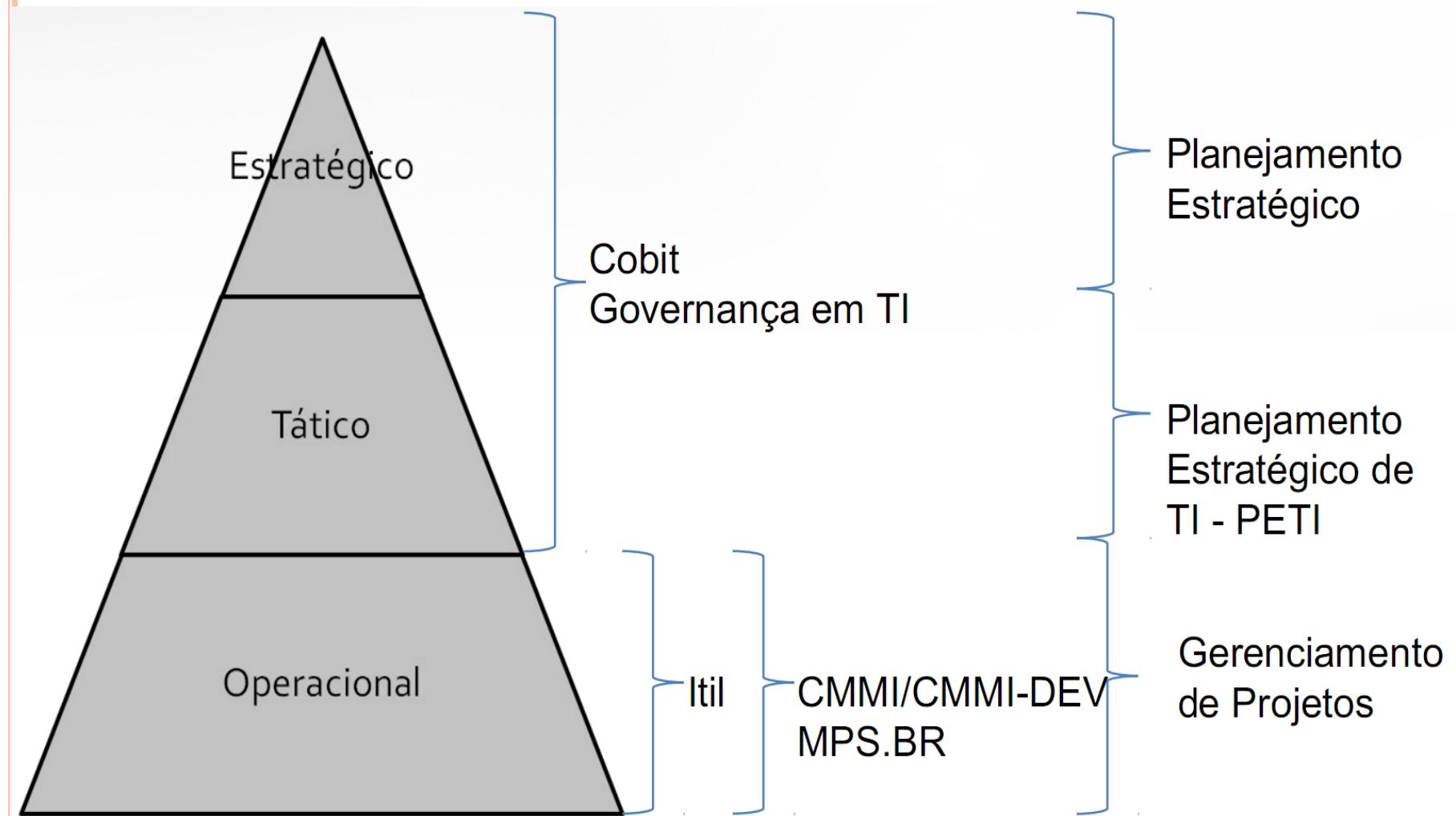
- Boas práticas do RUP

- Desenvolver o software iterativamente
 - Prioridade dos clientes junto com os recursos de alta prioridades, primeiro.
- Gerenciar requisitos
 - Documentar explicitamente os requisitos e acompanhar suas mudanças.
 - Avaliar uma mudança antes de aceitá-la.

RATIONAL UNIFIED PROCESS (RUP)

- Modelar o software visualmente
 - Utilizar modelos gráficos da UML para representar visões estáticas e dinâmicas do software.
- Verificar a qualidade de software
 - Assegurar que o software atende a qualidade organizacional.
- Controlar as mudanças do software

MPS.BR



MOTIVAÇÃO

- Em 2003, dados da Secretaria de Política de Informática do MCT apontavam que apenas 30 empresas no Brasil possuíam avaliação CMM e 214 possuíam certificação ISO 9001.
- Claramente, as empresas locais favoreceram a ISO 9000.
- Dados de uma pesquisa do MIT ¹, apontavam que até 2003, na Índia 32 empresas atingiram o nível 5 do CMM, enquanto a China tinha apenas uma e o Brasil nenhuma.
- Em relação ao CMM, a maioria das empresas chinesas e brasileiras não estava em um nível suficientemente alto de maturidade do processo para competir com as empresas indianas.

¹ Ref: *Slicing the Knowledge-based Economy in Brazil, China and India: a tale of 3 software industries* [MIT, 2003]

MOTIVAÇÃO: PROCESSO DE SOFTWARE NO BRASIL

EMPRESAS COM ISO 9000 E CMM

	1997	1999	2001	2003
Certificação ISO 9000	102	206	167	214
Avaliação CMM (total)	1	2	6	30
Nível 5	-	-	-	-
Nível 4	-	-	-	1
Nível 3	1	1	4	5
Nível 2	-	1	2	24

PROBLEMA DA EXCELÊNCIA: COMO ATINGIR NÍVEIS DE MATURIDADE CMMI NO BRASIL?

- No topo da pirâmide estão as empresas exportadoras de software e outras grandes empresas que desejam atingir níveis mais altos de maturidade (CMMI níveis 4 e 5) e serem formalmente certificadas pelo SEI, em um processo de longo prazo. O fator custo não é crítico.
- O processo como um todo pode levar de 4 a 10 anos e custar centenas de milhares de dólares.
- A melhoria de processo é baseada na oferta de serviços personalizados para cada empresa (Modelo de Negócio Específico).



PROBLEMA DA EXCELÊNCIA: COMO ATINGIR NÍVEIS DE MATURIDADE CMMI NO BRASIL?

Níveis de maturidade CMMI 4 e 5

Custo não é crítico – 4 a 10 anos

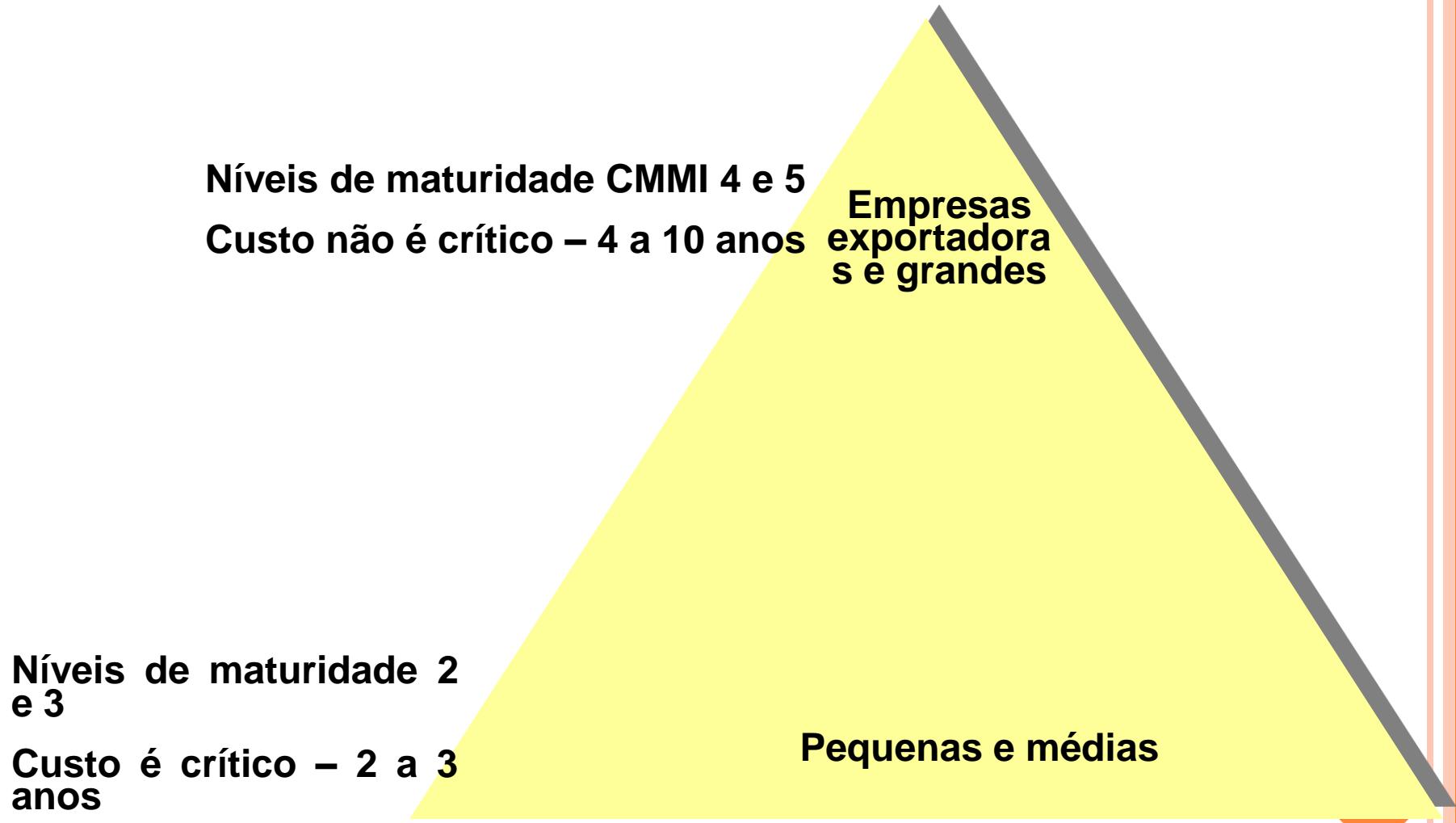
Empresas exportadoras e grandes

PROBLEMA DA EXCELÊNCIA: COMO ATINGIR NÍVEIS DE MATURIDADE CMMI NO BRASIL?

- Na base da pirâmide encontra-se a grande massa de **micro, pequenas e médias empresas (PMEs)** que desenvolvem software no Brasil e que necessitam melhorar radicalmente os seus processos de software, em conformidade com normas internacionais (como ISO/IEC 12207 e 15504) e em compatibilidade com outros modelos (como CMMI níveis 2 e 3). **O fator custo é crítico.**
- Esse processo pode levar de 2 a 4 anos e custar dezenas de milhares de dólares.
- A melhoria de processo deve ser baseada na oferta de pacotes de serviços para grupos de empresas (**Modelo de Negócio Cooperado**).



PROBLEMA DA EXCELÊNCIA: COMO ATINGIR NÍVEIS DE MATURIDADE CMMI NO BRASIL?



MPS.BR: OBJETIVO E METAS

- Objetivo: Melhoria de processos de software nas micros, pequenas e médias empresas (PMEs), a um custo acessível, em diversos locais do país.

Como?

- Desenvolvimento e Aprimoramento do Modelo MPS.BR.
- Implementação e Avaliação do Modelo MPS.BR em empresas, com foco em grupos de empresas.



MPS.BR: DESENVOLVIMENTO E APRIMORAMENTO

Base Técnica

ISO /IEC 12207

ISO /IEC 15504

CMMI

Realidade das Empresas Brasileiras



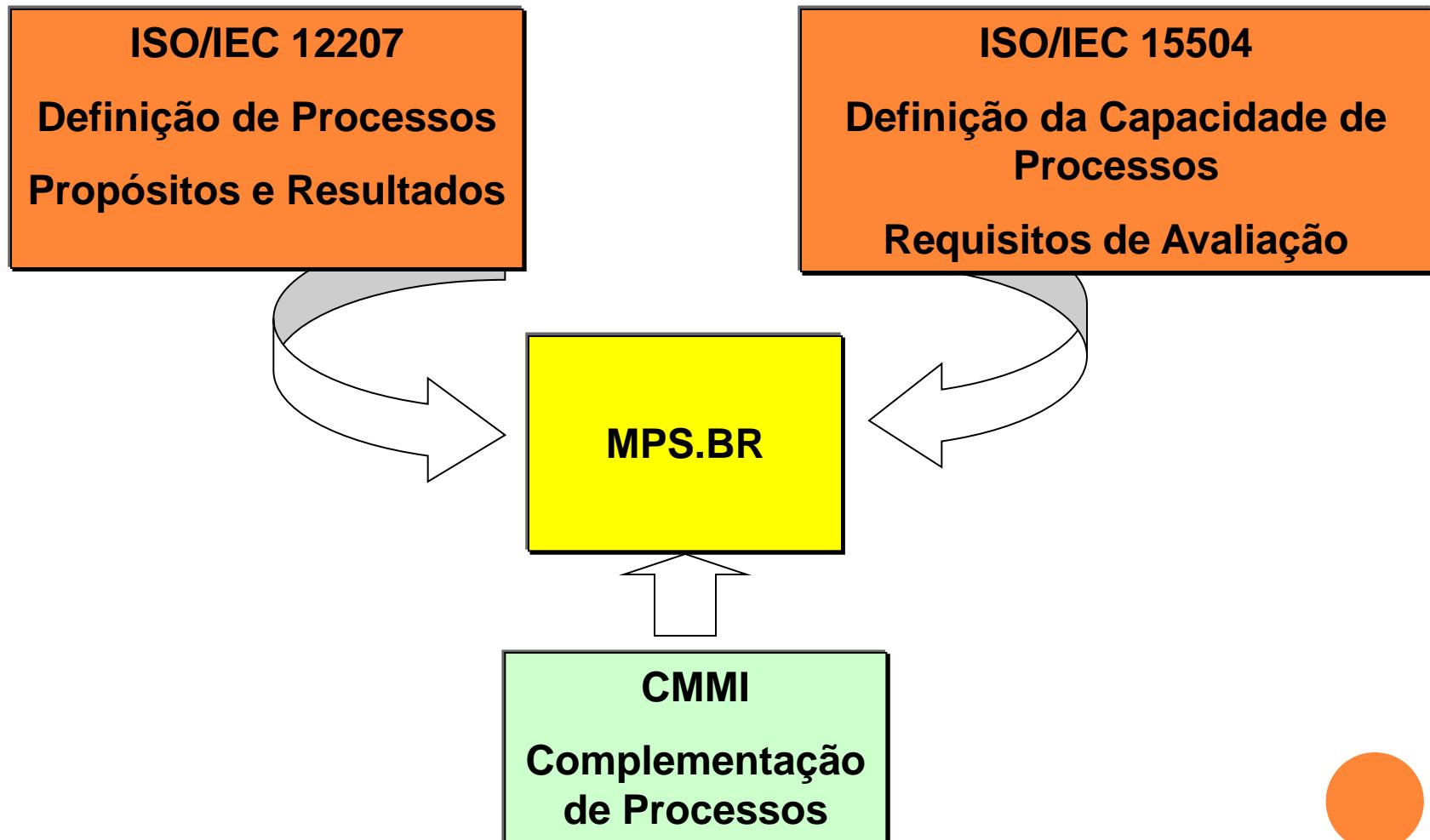
SOFTEX
Governo
Universidades



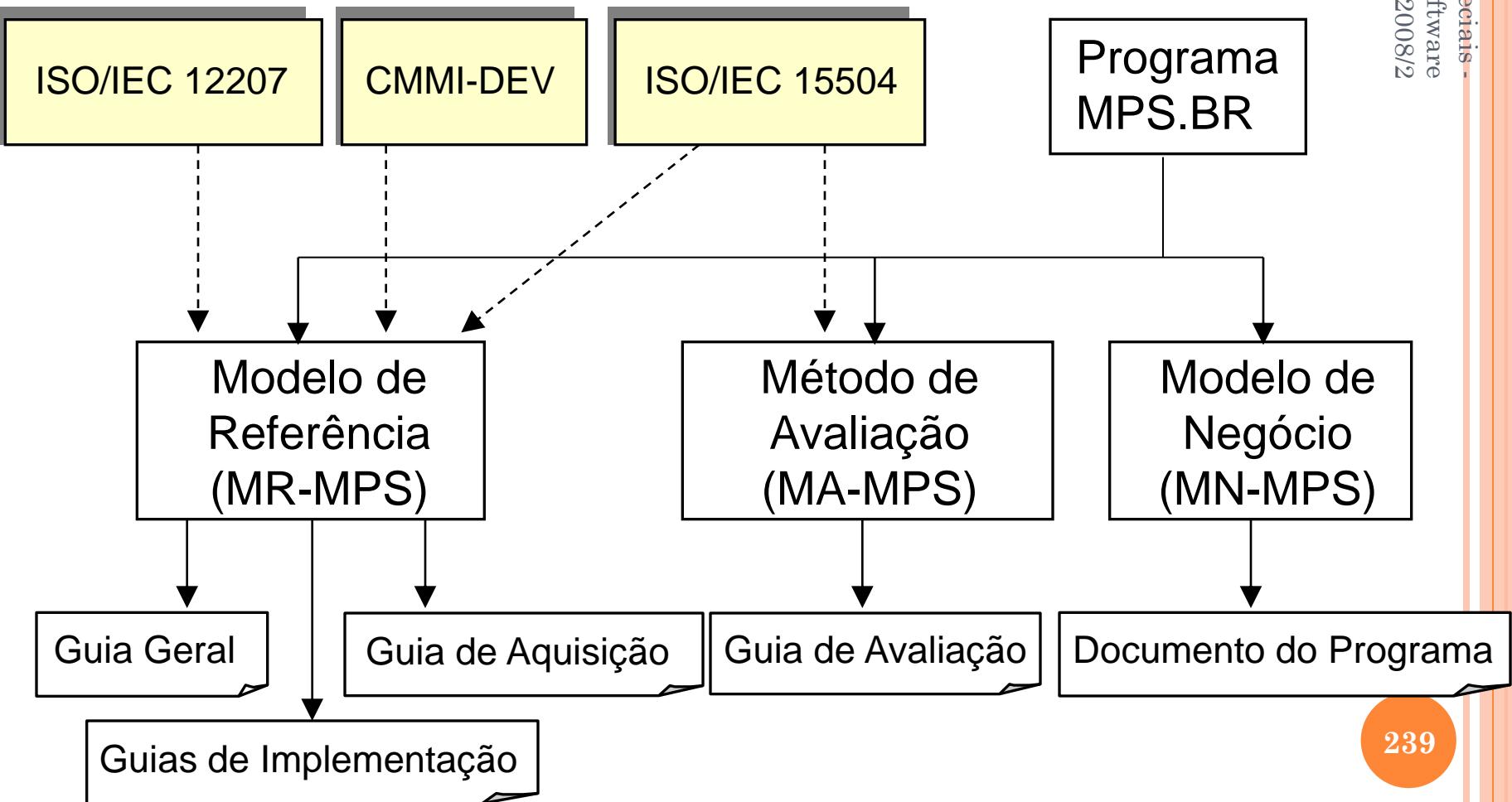
MPS.BR



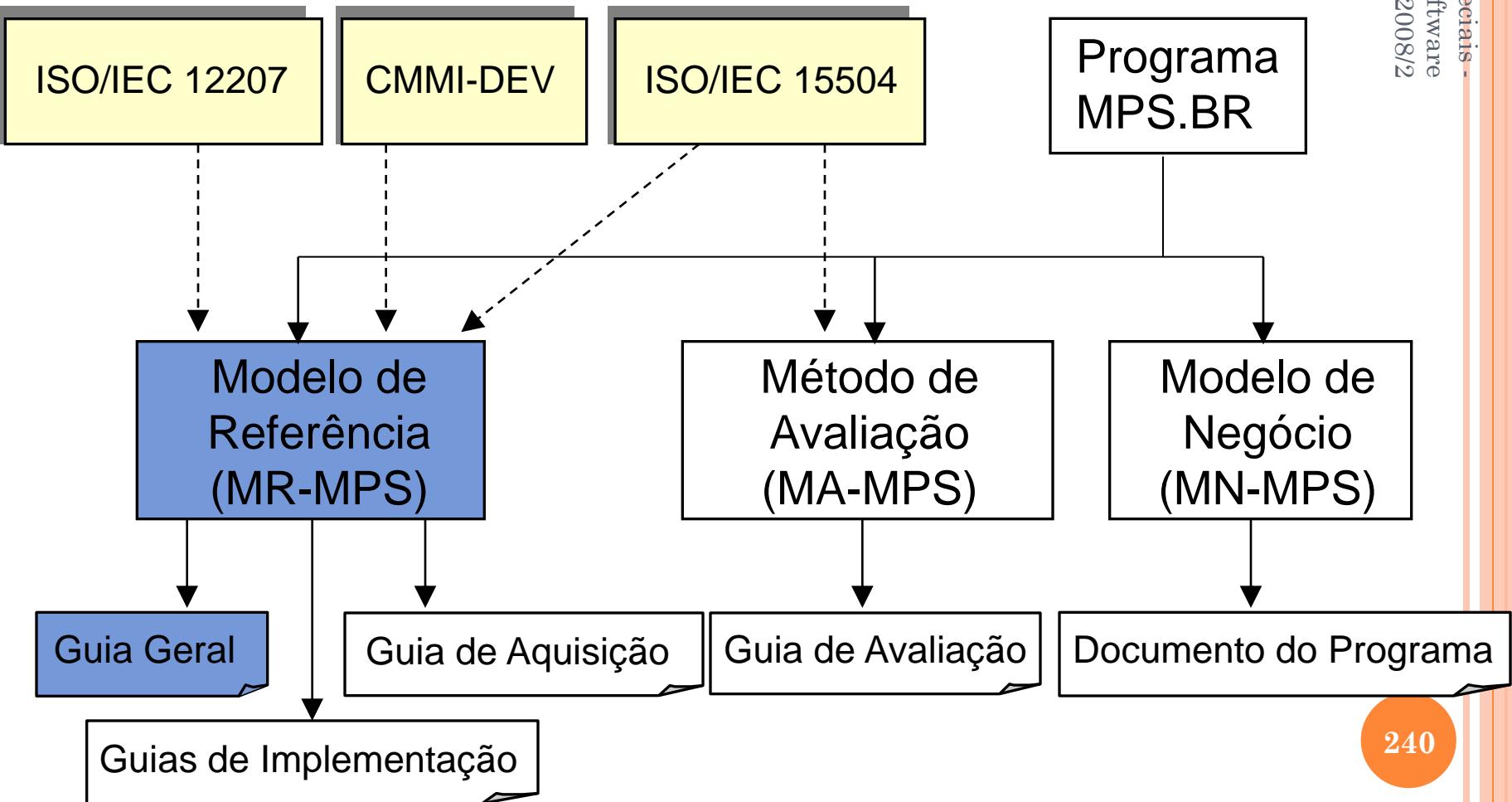
BASE TÉCNICA DO MPS.BR



ESTRUTURA DO MODELO MPS.BR



ESTRUTURA DO MODELO MPS.BR

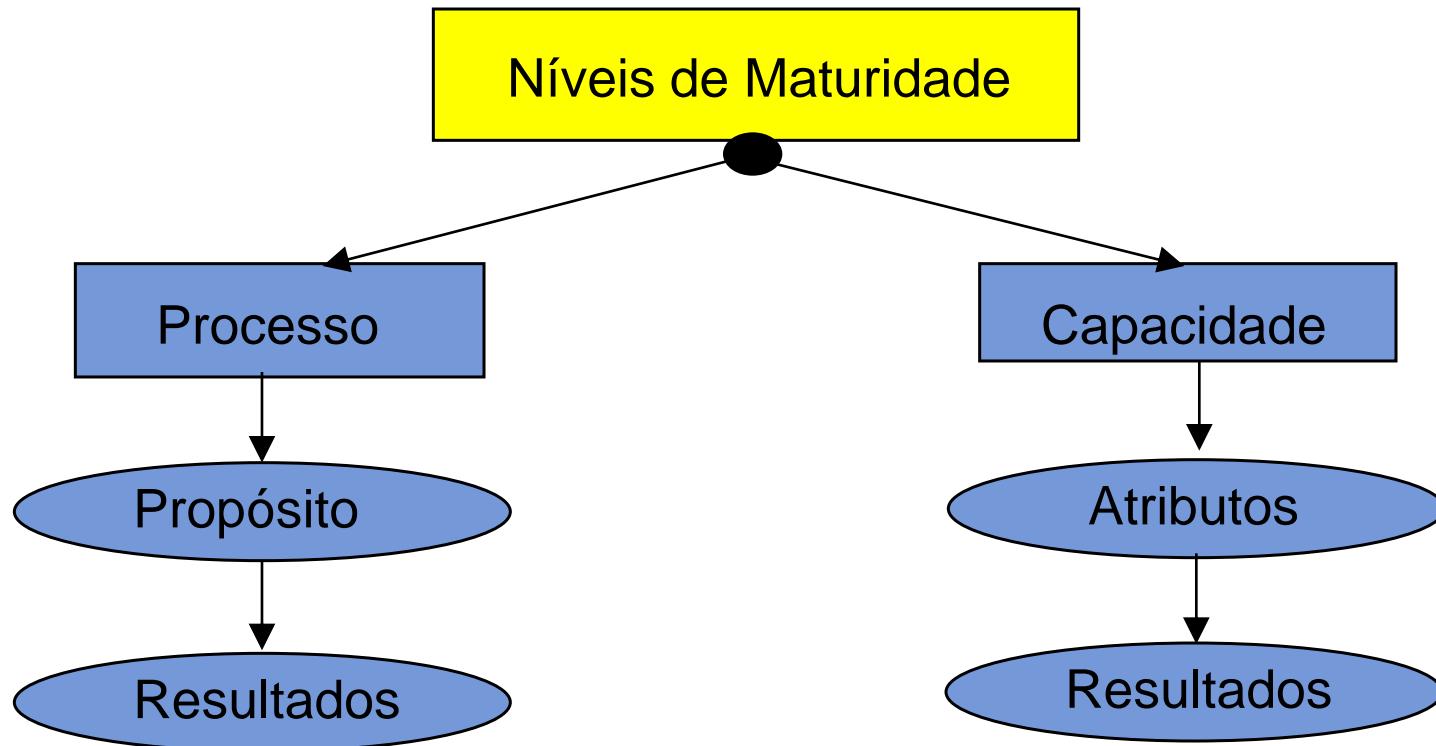


MPS.BR: GUIA GERAL

- Descreve o Modelo de Referência para Melhoria do Processo de Software (MR-MPS) e fornece uma visão geral sobre os demais guias que apóiam os processos de avaliação e de aquisição.
- Público-alvo:
 - Instituições interessadas em aplicar o MR-MPS para melhoria de seus processos de software.
 - Instituições Implementadoras (IIs) e avaliadoras (IAs) segundo o MR-MPS
 - outros interessados em processos de software e que pretendam conhecer e utilizar o MR-MPS como referência técnica.



ESTRUTURA DO MR-MPS



NÍVEIS DE MATURIDADE

- São uma combinação entre processos e sua capacidade.
- O progresso e o alcance de um determinado nível de maturidade do MR-MPS se obtém quando são atendidos os propósitos e todos os resultados esperados dos respectivos processos e dos atributos de processo estabelecidos para aquele nível.
- Os níveis de maturidade estabelecem patamares de evolução de processos, caracterizando estágios de melhoria da implementação de processos na organização.



NÍVEIS DE MATURIDADE

- O MR-MPS define sete níveis de maturidade:
 - A. Em Otimização
 - B. Gerenciado Quantitativamente
 - C. Definido
 - D. Largamente Definido
 - E. Parcialmente Definido
 - F. Gerenciado
 - G. Parcialmente Gerenciado



EQUIVALÊNCIA COM CMMI

Nível MPS.BR	Nível CMMI
G	
F	2
E	
D	3
C	
B	4
A	5



EQUIVALÊNCIA COM CMMI

- A divisão em estágios, embora baseada nos níveis de maturidade do CMMI-DEV, tem uma graduação diferente, com o objetivo de possibilitar uma implementação e avaliação mais adequada às micros, 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.

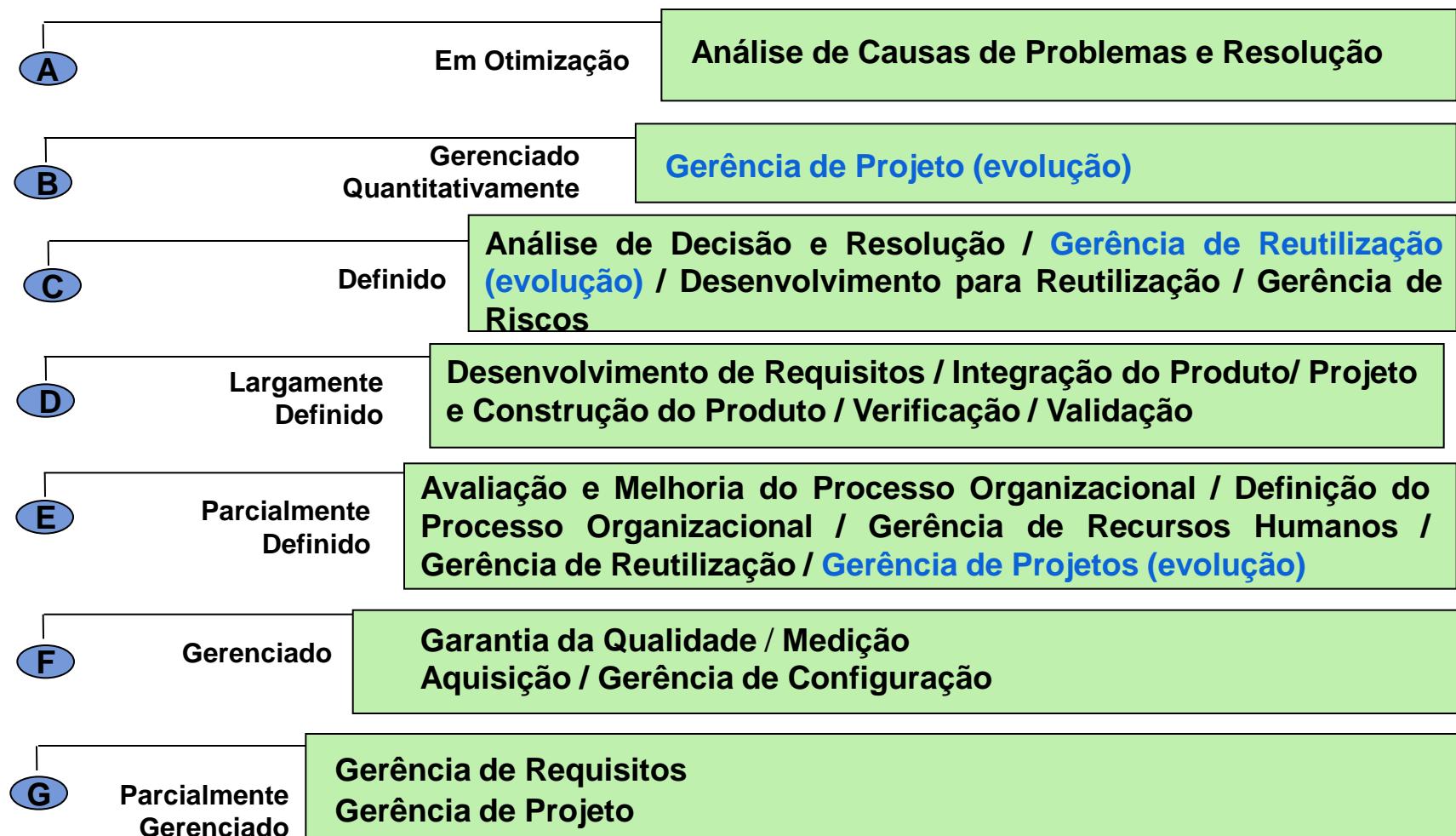


PROCESSO

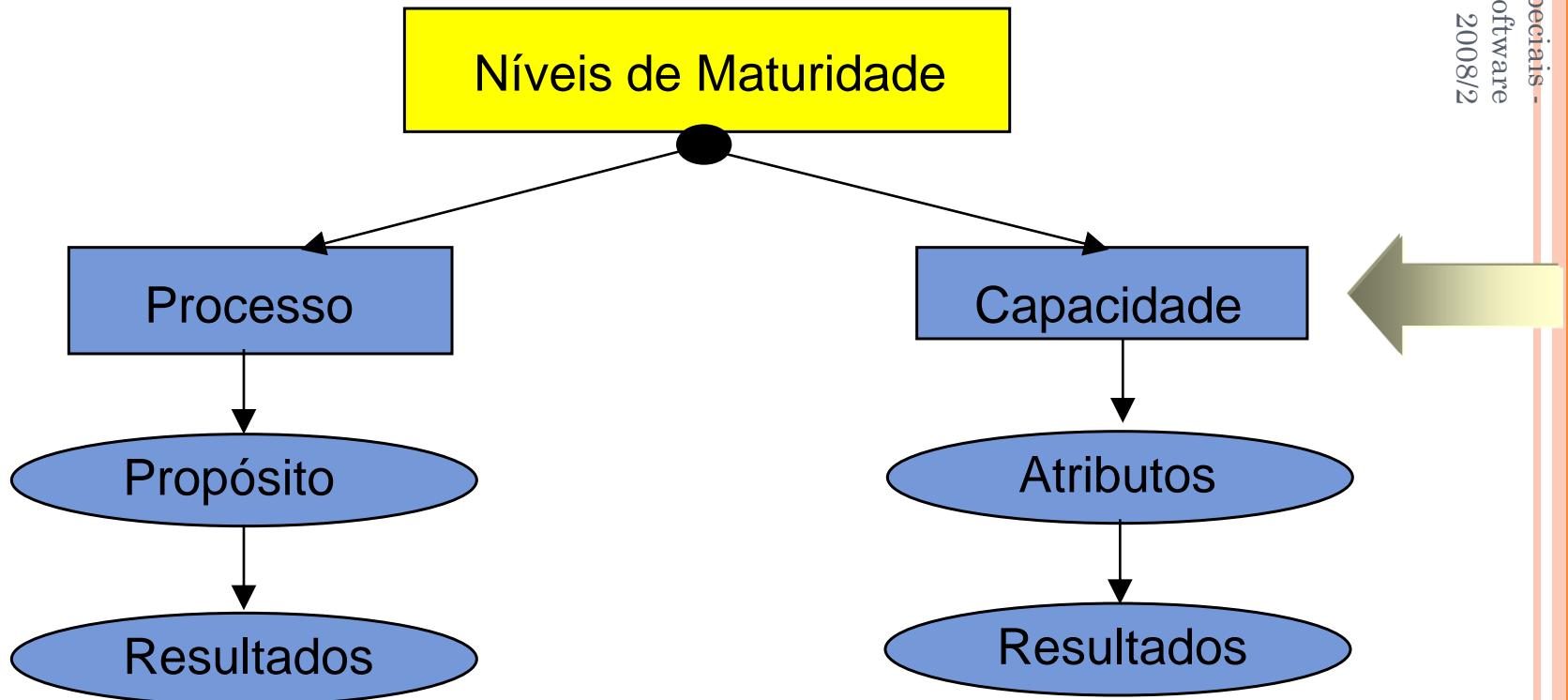
- Os processos no MR-MPS são descritos em termos de propósito e resultados.
 - **Propósito:** descreve o objetivo geral a ser atingido durante a execução do processo.
 - **Resultados Esperados:** estabelecem os resultados a serem obtidos com a efetiva implementação do processo. Esses resultados podem ser evidenciados por um artefato produzido ou uma mudança significativa de estado ao se executar o processo.



NÍVEIS DE MATURIDADE E PROCESSOS



ESTRUTURA DO MR-MPS



CAPACIDADE DO PROCESSO

- Expressa o grau de refinamento e institucionalização com que o processo é executado na organização / unidade organizacional.
- Está relacionada com o atendimento aos atributos de processo associados aos processos de cada nível de maturidade.
- À medida que a organização / unidade organizacional evolui nos níveis de maturidade, um maior nível de capacidade para desempenhar o processo deve ser atingido pela organização.



CAPACIDADE E ATRIBUTOS DE PROCESSO

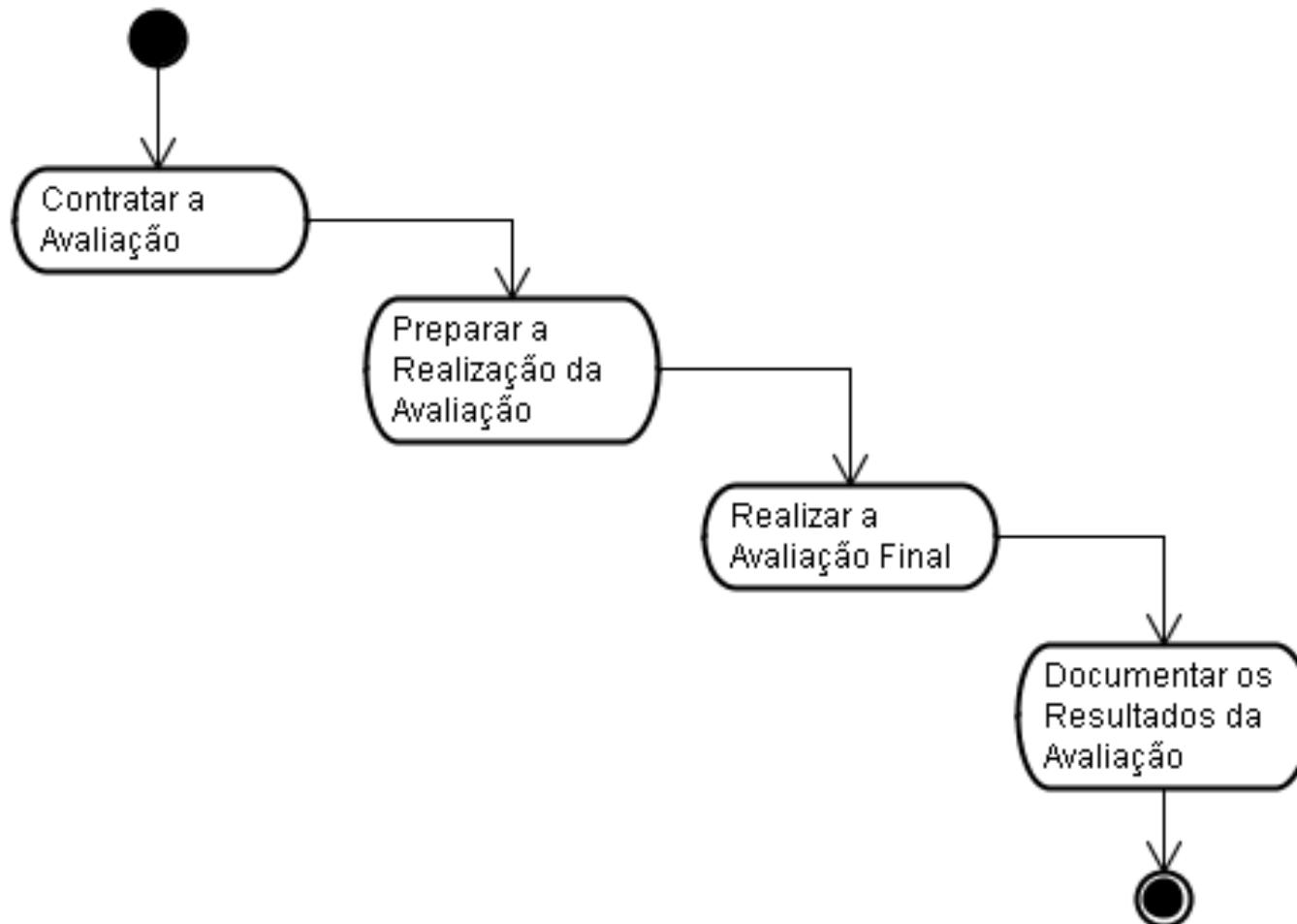
○ Atributos de Processo (AP):

- AP 1.1 – O processo é executado
- AP 2.1 – O processo é gerenciado
- AP 2.2 – Os produtos de trabalho do processo são gerenciados
- AP 3.1 – O processo é definido
- AP 3.2 – O processo está implementado
- AP 4.1 – O processo é medido
- AP 4.2 – O processo é controlado
- AP 5.1 – O processo é objeto de inovações
- AP 5.2 – O processo é otimizado continuamente

Introduzidos na Versão 1.2



O PROCESSO DE AVALIAÇÃO MPS.BR



EQUIPE DE AVALIAÇÃO

- No mínimo 3 pessoas:
 - 1 Avaliador Líder
 - 1 ou mais Avaliadores Adjuntos
 - 1 ou mais Representante da Unidade Organizacional
 - Deve ter assistido ao Curso Oficial de Introdução ao MPS.BR.
 - Deve ter experiência em desenvolvimento de software, preferencialmente em gerência de projetos
 - Não pode ser superior hierárquico dos participantes da avaliação
 - Não pode ter tido participação significativa em nenhum dos projetos que serão avaliados



TAMANHO DA EQUIPE DE AVALIAÇÃO

Níveis	Nº min – Nº max
A e B	8 - 9
C e D	6 - 7
E e F	4 - 5
G	3 - 4



REGRAS PARA CARACTERIZAR O GRAU DE IMPLEMENTAÇÃO DOS ATRIBUTOS DE PROCESSO NA UO

Grau de implementação	Caracterização	Porcentagem de Implementação dos resultados relacionados ¹⁵
Totalmente implementado (T)	Existe evidência de um enfoque completo e sistemático para o atributo no processo avaliado e de sua plena implementação. Não existem pontos fracos relevantes para este atributo no processo avaliado	>85% a 100%
Largamente implementado (L)	Existe evidência de um enfoque sistemático e de um grau significativo de implementação do atributo no processo avaliado. Existem pontos fracos para este atributo no processo avaliado	>50% a 85%
Parcialmente implementado (P)	Existe alguma evidência de um enfoque para o atributo e de alguma implementação do atributo no processo avaliado. Alguns aspectos de implementação não são possíveis de predizer.	>15% a 50%
Não implementado (N)	Existe pouca ou nenhuma evidência de implementação do atributo no processo avaliado	0 a 15%

CARACTERIZAÇÃO DO GRAU DE IMPLEMENTAÇÃO DE CADA UM DOS PROCESSOS

- Um processo está SATISFEITO quando:
 - Todos os resultados esperados para o processo foram caracterizados como T (Totalmente Implementado) ou L (Largamente Implementado).
 - Tem-se resultados para os atributos do processo, conforme a tabela a seguir.
- Em qualquer outra situação o processo é caracterizado como NÃO SATISFEITO.



DIFERENCIAIS DO MPS.BR

- 7 níveis de maturidade, o que possibilita uma implantação mais gradual e uma maior visibilidade dos resultados de melhoria de processo, com prazos mais curtos.
- Compatibilidade com CMMI, conformidade com as normas ISO/IEC 15504 e 12207.
- Adaptado para a realidade brasileira (foco em micro, pequenas e médias empresas).
- Custo acessível (em R\$)



REFERÊNCIAS

- Softex, MPS.BR - Melhoria de Processo do Software Brasileiro – Guia Geral.
- Softex, MPS.BR - Melhoria de Processo do Software Brasileiro – Guia de Avaliação.
- Empresas: http://www.softex.br/mpsbr/_avaliacoes/default.asp



OS 10 MANDAMENTOS DO PROCESSO IMATURO

- Os slides a seguir apresentam os 10 mandamentos de um processo imaturo. Logo, ele relata o que fazer para se ter um processo imaturo. Ou seja, é abordado o que **NÃO** fazer.



OS 10 MANDAMENTOS DO PROCESSO IMATURO

- ***10º: Não estabelecer métricas para o desenvolvimento de software***
- Cada software é desenvolvido de uma forma particular, em função das suas características, e também cada desenvolvedor tem um estilo próprio de codificação. Assim não é possível nem necessário estabelecer métricas como produtividade por linha de código, quantidade de erros por pontos de função detectados em ambiente de produção, cumprimento dos prazos de desenvolvimento, etc. A equipe, mesmo sem métricas tende a melhorar organicamente.



OS 10 MANDAMENTOS DO PROCESSO IMATURO

- **9º: *Não prever capacitação dos usuários para utilização do software***
- Atualmente as interfaces gráficas são muito intuitivas e de fácil utilização. As crianças já utilizam computadores desde a tenra idade e crescem em contato com esse ambiente. Como hoje os prazos de desenvolvimento são apertados não se faz necessário gastar tempo com a capacitação dos usuários para a utilização do software. Havendo dúvidas de utilização, o usuário sempre pode ligar para o *help-desk*.

OS 10 MANDAMENTOS DO PROCESSO IMATURO

- ***8º: Não utilizar um processo definido para relato de defeitos***
- Durante o processo de desenvolvimento e as diversas etapas de testes, à medida que os defeitos vão sendo encontrados, eles vão sendo priorizados e corrigidos segundo a própria experiência dos programadores. Não há necessidade de um software para gestão de defeitos, o que iria só burocratizar a agilidade da correção dos mesmos. Raramente ocorre de um software ir para a produção com defeitos já conhecidos, que foram esquecidos de serem consertados pelos programadores.

OS 10 MANDAMENTOS DO PROCESSO IMATURO

- **7º: *Não utilizar um software de controle de versão***
- Os códigos fontes são mantidos nas máquinas dos programadores envolvidos em cada projeto de desenvolvimento, de forma a dar maior liberdade e velocidade ao programador. O que importa é a experiência e o controle efetuado pelo programador na hora de colocar o software em produção. Um software de controle de versão é caro e burocrático.

OS 10 MANDAMENTOS DO PROCESSO IMATURO

- *6º: Definir a arquitetura do software à medida que o código vai ficando pronto*
- Pensar e desenhar a arquitetura do software antes do código estar pelo menos 60 a 80 % pronto não é produtivo, e acaba sendo uma atividade de abstração que quase sempre se demonstra inútil. O programador de acordo com a necessidade do código vai definindo a arquitetura necessária e assim o resultado é sempre um software com boa funcionalidade, usabilidade e performance.

OS 10 MANDAMENTOS DO PROCESSO IMATURO

- **5º: *Afastar o cliente do processo de desenvolvimento***
- O desenvolvimento de software é de competência exclusiva de analistas e programadores, assim uma vez que já se obteve uma descrição funcional do software a ser desenvolvimento não se faz mais necessário a participação do cliente no processo de desenvolvimento. Com o afastamento do cliente a equipe de desenvolvimento se mantém mais focada, o software tem mais chances de ser entregue no prazo e de acordo com as necessidades do cliente.

OS 10 MANDAMENTOS DO PROCESSO IMATURO

- **4º: *Não utilizar uma equipe de teste independente***
- As atividades de testes são onerosas em termos de custos e prazos, assim não se faz necessário a utilização de uma equipe independente de testes. Na maioria das vezes os testes realizados pelo próprio programador garantem um software de boa qualidade, entregue no prazo e com custos controlados.



OS 10 MANDAMENTOS DO PROCESSO IMATURO

- **3º: Utilizar o programador cowboy: aquele que faz todo o desenvolvimento do software sozinho**
- A divisão de papéis na equipe de desenvolvimento, como analista de negócios, analista de requisitos, arquiteto, programador e testador só burocratiza o processo de desenvolvimento sem trazer benefícios relevantes para a qualidade do software. Assim utilizar apenas um profissional desempenhado todos esses papéis resulta sempre em melhores resultados.

OS 10 MANDAMENTOS DO PROCESSO IMATURO

- **2º: *Codificar antes de especificar***
- Iniciar a codificação do software o mais rápido possível, ainda que os requisitos não tenham sido claramente definidos torna o processo de desenvolvimento mais ágil, permite ao cliente ter uma melhor noção do que ele precisa, além de garantir entregas mais rápidas e de melhor qualidade.

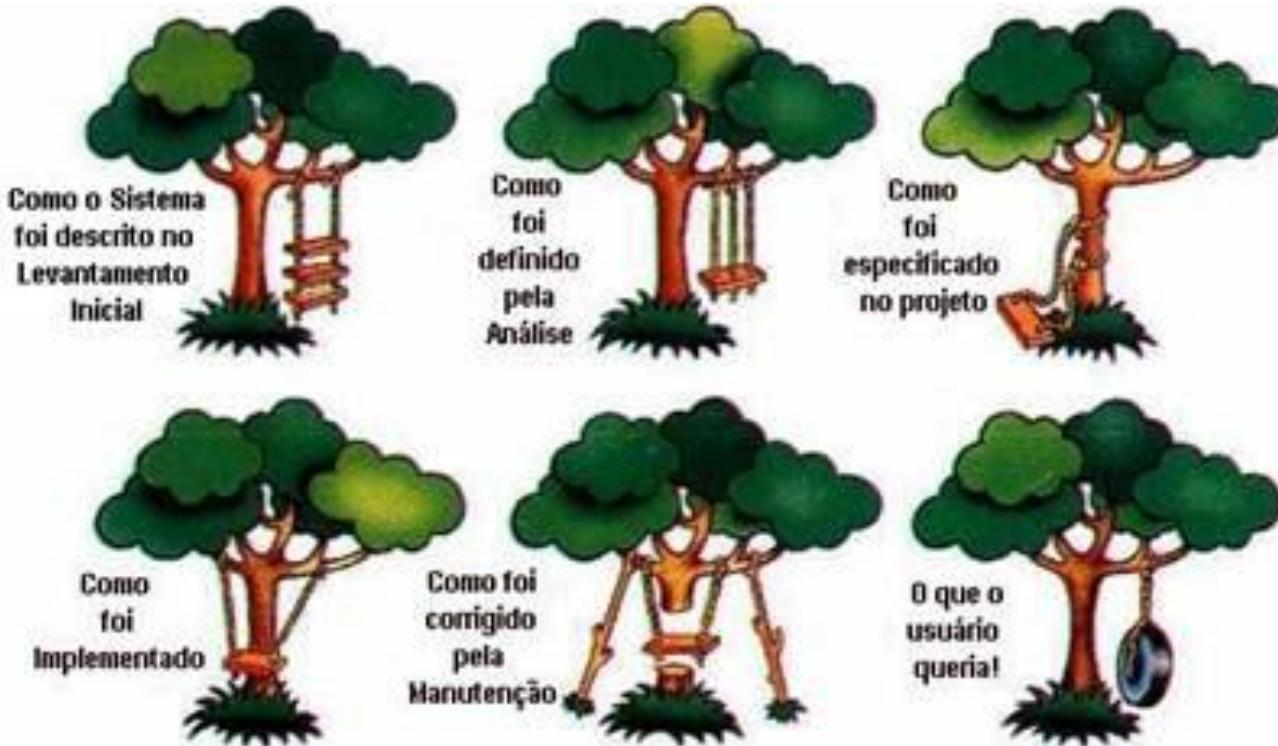


OS 10 MANDAMENTOS DO PROCESSO IMATURO

- **1º: *Estabelecer cronograma irreal***
- Atualmente em função das demandas de mercado, os cronogramas de desenvolvimento de software devem ser agressivos, ainda que pareçam irreais. Sempre é possível aumentar a equipe de desenvolvimento, reduzir prazos com atividades de arquitetura e testes, e em último caso renegociar o prazo com o cliente.



QUALIDADE DE SOFTWARE



QUALIDADE DE SOFTWARE

- Conferência da NATO (1968) – Crise de Software
- Problemas detectados:
 - Cronogramas não observados.
 - Projetos abandonados.
 - Módulos que não operam corretamente quando combinados.
 - Programas que não fazem exatamente o que era esperado.
 - Sistemas tão difíceis de usar que são descartados.
 - Sistemas que simplesmente param de funcionar.
- Passados 40 anos, o que mudou?

QUALIDADE DE SOFTWARE

- Os 10 maiores bugs da história, além do bug do milênio:
 - **28 de Julho de 1962: Falha na sonda Mariner 1.** Um bug no software de vôo da sonda Mariner 1 provocou que, segundos após o lançamento da nave, esta se desviasse de seu curso pré-estabelecido. Os responsáveis da missão foram obrigados a destruir o foguete quando se encontrava sobrevoando o Atlântico.
A investigação do acidente determinou que o problema estava numa fórmula escrita a lápis que depois foi "inadequadamente" digitada no computador de cálculo, o que fez que o foguete calculasse mal a trajetória que devia seguir.

QUALIDADE DE SOFTWARE

- **1982: Explosão num gasoduto soviético.**

A maior explosão registrada na Terra por causas não nucleares teve sua origem numa falha de programação. Supostamente, agentes da CIA infiltraram um bug num sistema de informática Canadense adquirido pelos soviéticos para controlar o gaseoduto Transiberiano. Seguiam ordens de Reagan, que tinha mandado seus agentes sabotar toda a tecnologia russa, colocando artefatos que permitissem manipular a distância todo tipo de máquinas e tecnologia. Assim, em 1982 a CIA decidiu sabotar este gasoduto, mas ao ativar o bug as coisas saíram muito diferente do esperado provocando a gigantesca explosão.

QUALIDADE DE SOFTWARE

- **1985-1987: Acelerador médico Therac-25.**

O Therac-25 era um acelerador linear empregado nos hospitais na década de 80 para tratar tumores. A máquina emitia radiação de alta energia sobre células cancerosas sem causar dano ao tecido circundante. Os funcionários, com o tempo e a prática, conseguiam grande velocidade digitando a sequência de comandos para iniciar um tratamento. Mas devido a uma falha de programação, durante um processo onde efetuavam estas correções, a máquina emitia 100 vezes mais energia do que a requerida. Em consequência deste bug morreram ao menos cinco pacientes e várias dezenas sofreram os efeitos de ficarem expostos a uma elevada radiação, inclusive os próprios funcionários.

QUALIDADE DE SOFTWARE

- **1988: O Worm de Morris.**

O primeiro vírus da Internet nasceu na tarde de 2 de novembro de 1988, quando um estudante norte-americano, Tappan Morris, liberou um programa criado por ele mesmo que infectou entre 2.000 e 6.000 computadores só no primeiro dia, antes de ser rastreado e eliminado. Para que seu vírus tivesse efeito, Morris descobriu dois erros no sistema operacional UNIX, que lhe permitiram ter acesso não autorizado a milhares de computadores.

QUALIDADE DE SOFTWARE

- **1988-1996: Gerador de números aleatórios de Kerberos**

Os autores do sistema de geração de números aleatórios Kerberos que são utilizados para fazer comunicações seguras através da Rede falharam à hora de conseguir que seu programa realmente escolhesse os números aleatoriamente. Devido a essa falha, durante oito anos foi possível entrar em qualquer computador que utilizasse o sistema Kerberos para autentificação, ainda que realmente se desconhece se o bug chegou a ser aproveitado por alguém.

QUALIDADE DE SOFTWARE

- **15 de Janeiro de 1990: Queda da rede de AT&T.**

Um bug no software que controlava os comutadores dos telefonemas de longa distância da gigante da telefonia ATT fazia que derrubasse a chamada no comutador vizinho quando recebiam uma determinada mensagem. Por essa falha, o comutador da cidade de Nova York derrubou outras centenas de comutadores causando um caos na telefonia daquele país.

QUALIDADE DE SOFTWARE

- **1993: Divisão de números com ponto flutuante no Pentium.**

Um problema com os microprocessadores provocou uma falha na divisão de números com ponto flutuante. Por exemplo, ao dividir 4195835,0 por 3145727,0 o resultado apresentado pelo microprocessador era 1,33374 ao invés de 1,33382, um erro de 0.006%. Ainda que a falha afetava a poucos usuários, resultou todo um problema para a Intel, que viu-se obrigada a trocar entre três e cinco milhões de chips, numa operação que lhe custou mais de meio bilhão de dólares.

QUALIDADE DE SOFTWARE

- **1995/1996: O Ping da Morte.**

Devido a um problema que afetava o código que maneja o protocolo IP, era possível "capturar" um computador com Windows lhe enviando um ping corrupto. O problema afetava vários sistemas operacionais mas o pior caso era, lógico, com o Windows, que travava e mostrava a famosa "tela azul".

QUALIDADE DE SOFTWARE

- **4 de Junho de 1996: Desintegração do Ariane 5.**

Os cientistas que desenvolveram o foguete Ariane 5, voo 501, reutilizaram parte do código de seu predecessor, o Ariane 4, mas os motores do novo foguete incorporavam também, sem que ninguém desse conta, um bug numa rotina aritmética no computador de voo que falhou segundos após a decolagem do foguete; em decorrência, meio segundo depois o computador principal da missão também apresentou problemas. O Ariane 5 desintegrhou-se 40 segundos após o lançamento.

QUALIDADE DE SOFTWARE

- **Novembro 2000: Sobredosagem radiológica no Instituto Nacional do Cancro da Cidade do Panamá**

Numa série de acidentes, falhas e verdadeiras trapalhadas em seqüência, os engenheiros da empresa Multidata Systems International calcularam erroneamente a dose de radiação que um paciente deveria receber durante a terapia de radiologia. A falha estava no software de controle da máquina de raios, que provocou que ao menos oito pacientes morressem pelas altas doses recebidas e outros 20 recebessem sobredosagens que poderiam causar graves danos a sua saúde

QUALIDADE DE SOFTWARE

- “A qualidade de software é um conjunto de características ou fatores de software, que determinam o nível de eficiência do software em uso, em relação ao atendimento das expectativas dos clientes”. (IEEE).
- “Conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido” (Pressman)

QUALIDADE DE SOFTWARE

○ Dificultadores:

- O aspecto não repetitivo do desenvolvimento de software torna essa atividade difícil e em boa medida imprevisível.
- Delimitar o escopo de um sistema não é trivial. A volatilidade dos requisitos é lugar comum no desenvolvimento de software.

QUALIDADE DE SOFTWARE

- Motivação para a busca da Qualidade do Processo de Software:
 - Aumento da qualidade do produto.
 - Diminuição do retrabalho.
 - Maior produtividade.
 - Redução do tempo para atender o mercado (time to market).
 - Maior competitividade.
 - Maior precisão nas estimativas

QUALIDADE DE SOFTWARE

- O que o cliente quer?

- Atendimento aos requisitos especificados
- Defeito zero
- Alto desempenho
- Baixo custo
- Desenvolvimento rápido
- Facilidade de uso
- Eficiência nos serviços associados
- Inovação

QUALIDADE DE SOFTWARE

- Para que um software tenha qualidade ele deve:
 - Preencher as expectativas do cliente;
 - Ser obtido dentro de um prazo previsto;
 - Ser produzido dentro de custos pré-estabelecidos;
 - Conformar com as especificações de requisitos previamente estabelecidas

QUALIDADE DE SOFTWARE

- Para a obtenção de um software com qualidade, deve-se:
 - Definir claramente o seu objetivo, a sua finalidade, o seu propósito;
 - Especificar seus requisitos para atender as necessidades do usuário;
 - Produzi-lo e utilizá-lo dentro de processos bem estabelecidos.

FATORES DE QUALIDADE DE SOFTWARE

- Explícitos – visíveis para o usuário
 - Usabilidade – Expressa a facilidade de uso
 - Confiabilidade – Capacidade de dependência do software, por determinado período de tempo
 - Integridade – Controle de acesso ao sistema
 - Prazo – Prazo estimado de entrega
 - Informações sobre o progresso – Relatórios descrevendo o progresso
 - Tempo de atendimento – Tempo gasto para as manutenções
 - Retorno do Investimento – Retorno em forma de benefícios

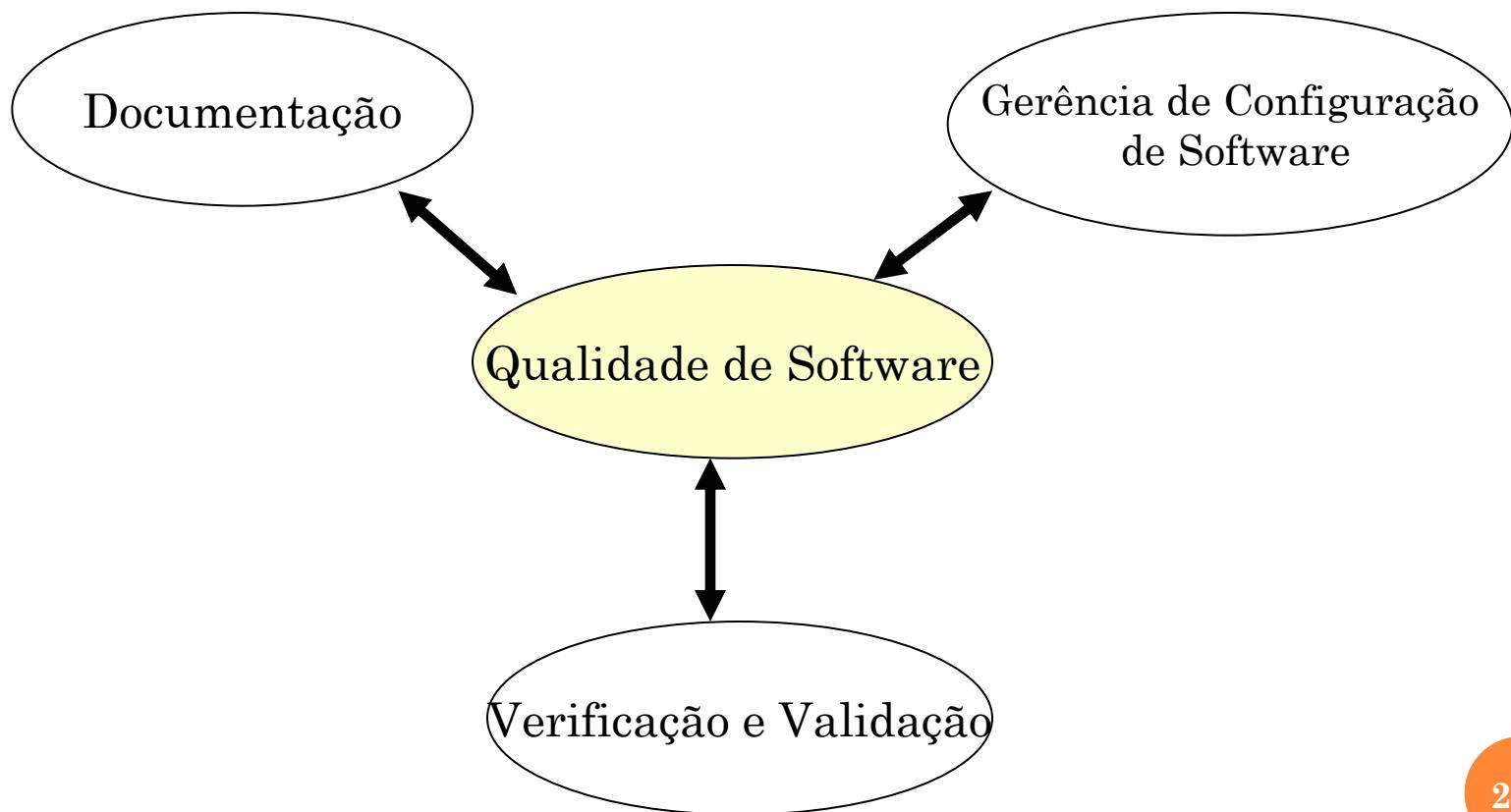
FATORES DE QUALIDADE DE SOFTWARE

- Implícitos – visíveis para os desenvolvedores
 - Flexibilidade – facilidade de modificação
 - Manutenabilidade – esforço necessário para remover defeitos
 - Testabilidade – Facilidade de execução de testes
 - Eficiência – quantidade de recursos para cumprir determinada tarefa
 - Interoperabilidade – Integração das partes de um sistema
 - Reusabilidade – Possibilidade de reaproveitamento de software/partes
 - Portabilidade – Capacidade de usar diferentes plataformas
 - Estimativas – Exatidão nas estimativas de custo/prazo/esforço
 - Estabilidade – Extensão do ciclo de vida onde ele mantém a qualidade

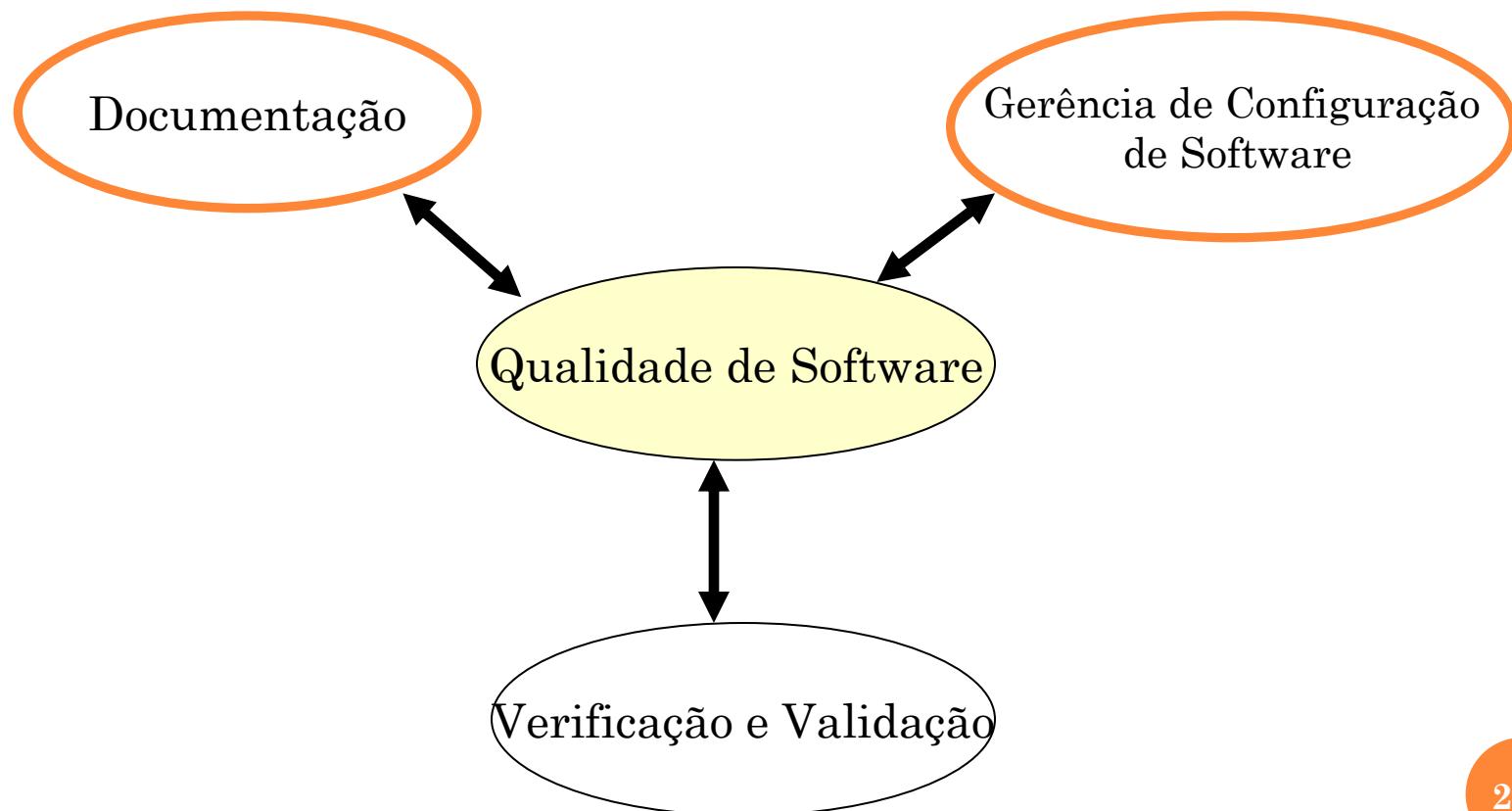
QUALIDADE DE SOFTWARE

- Qualidade no Processo de Desenvolvimento do Software
 - Definir um processo adequado para o ciclo de desenvolvimento;
 - Selecionar e aplicar métodos adequados de análise, projeto e implementação;
 - Definir processos adequados de verificação e validação (testes);
 - Sistematizar os testes por meio de planos, procedimentos e documentos de teste;
 - Utilizar ferramentas adequadas;
 - Aplicar normas e padrões pertinentes;
 - Gerenciar a configuração do software;
 - Acompanhar e avaliar a evolução das especificações de requisitos.

QUALIDADE E PROCESSOS RELACIONADOS



QUALIDADE E PROCESSOS RELACIONADOS



DOCUMENTAÇÃO E GERÊNCIA DE CONFIGURAÇÃO

- Artefatos registram a evolução do software para que sejam criadas as bases para o desenvolvimento, utilização e manutenção efetivos.
- Artefatos devem retratar fielmente o software, de modo que as atividades de avaliação e modificação possam ser realizadas sem maiores transtornos.
- Artefatos evidenciam a evolução do projeto. Mas é muito importante registrar modificações que ocorrem nos mesmos, de modo a se ter um histórico da evolução, o que é feito por meio da Gerência de Configuração de Software (GCS).

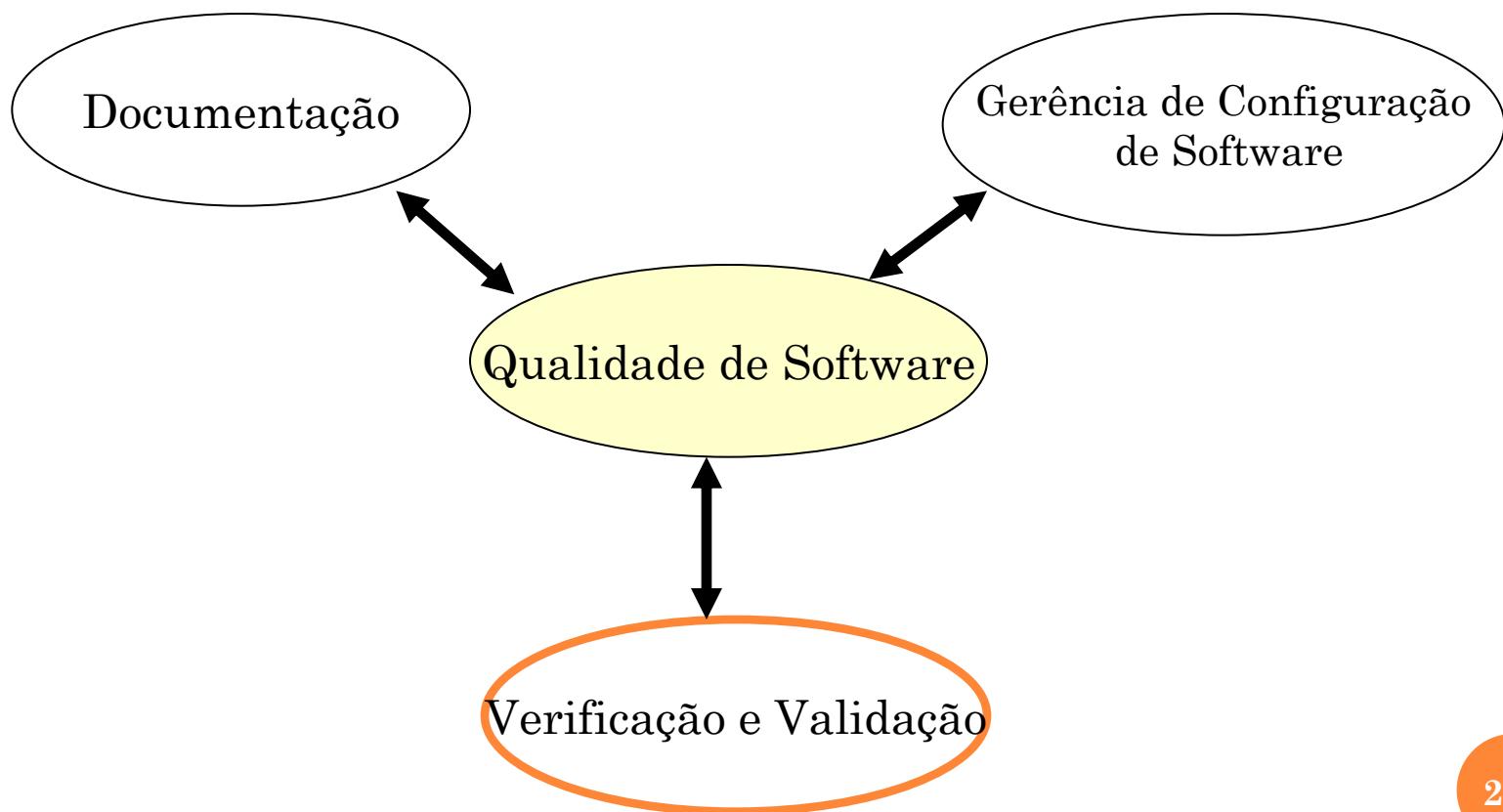


GERÊNCIA DE CONFIGURAÇÃO

- Permite manter o controle da evolução dos artefatos de software, além de ajudar a cumprir metas de garantia da qualidade.
- Envolve, dentre outros:
 - controle de alterações,
 - registro e apresentação da situação dos itens e das solicitações de alteração,
 - garantia da consistência dos itens alterados,
 - a identificação de itens de configuração de software,
 - controle de versão, armazenamento, manipulação e distribuição de itens.



QUALIDADE E PROCESSOS RELACIONADOS



VERIFICAÇÃO E VALIDAÇÃO

- Verificação: assegurar que o software, ou determinada função do mesmo, está sendo desenvolvido corretamente, o que inclui verificar se os métodos e processos estão sendo aplicados adequadamente.

“Estamos construindo o produto correto?”

- Validação: assegurar que o software que está sendo desenvolvido é o software correto.

“Estamos construindo o produto corretamente?”



VERIFICAÇÃO E VALIDAÇÃO

- Depende do propósito do sistema, das expectativas dos usuários e do ambiente de mercado.
 - **Função de software**
 - O nível de confiança depende de quão crítico é o software para uma organização.
 - **Expectativas do usuário**
 - Os usuários podem ter baixas expectativas de certos tipos de software.
 - **Ambiente de mercado**
 - Colocação de um produto para o mercado mais cedo pode ser mais importante que descobrir defeitos no programa.

ANÁLISE ESTÁTICA E ANÁLISE DINÂMICA

- Análise Estática: não envolve a execução propriamente dita do produto. Pode e deve ser aplicada em qualquer artefato intermediário. Ex.: Revisões técnicas, inspeção de código.
- Análise Dinâmica: envolve a execução do produto. Ex.: Testes.



TESTES DE SOFTWARE

- “Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. O seu objetivo é revelar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final. Por conta dessa característica das atividades de teste, dizemos que sua natureza é “destrutiva”, e não “construtiva”, pois visa ao aumento da confiança de um produto através da exposição de seus problemas, porém antes de sua entrega ao usuário final.”

Leia mais em: [Artigo Engenharia de Software - Introdução a Teste de Software](http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035#ixzz2C8TQWBGv) <http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035#ixzz2C8TQWBGv>

POR QUE TESTAR?

- Você conhece as leis de Murphy?
 - Primeira lei - Se uma coisa pode sair errado, sairá.
 - Se tudo parece estar indo bem, é porque você não olhou direito.
 - A natureza sempre está a favor da falha oculta

POR QUE TESTAR?

- "O teste consiste em executar o programa com a intenção de encontrar erros (bugs)". **Myers, 1979**
- Ao testarmos um software, estamos buscando:
 - Prevenir erros em fases futuras (mais custosa)
 - Desvendar sintomas causados por erros
 - Fornecer diagnósticos para que a causa raiz dos bugs seja atacada

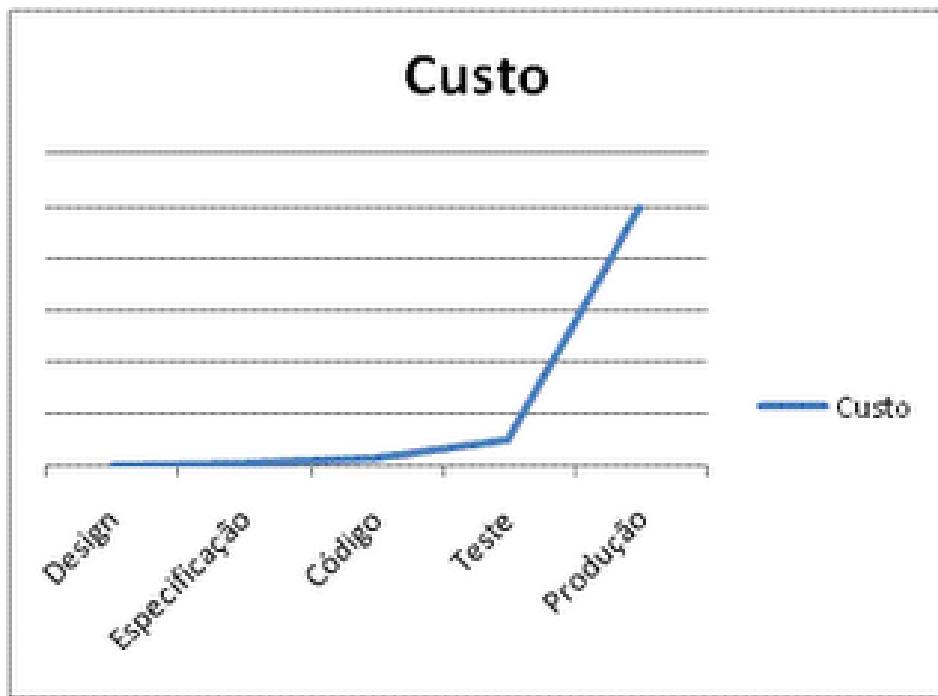
TESTAR É FÁCIL?

- Nem tanto:

- Os principais erros não são óbvios;
- Encontrar o erro não significa encontrar a solução;
- Erros totalmente distintos podem se manifestar de forma semelhante.

CUSTO DE UM BUG

O objetivo principal desta tarefa é revelar o número máximo de falhas dispondendo do mínimo de esforço, ou seja, mostrar aos que desenvolvem se os resultados estão ou não de acordo com os padrões estabelecidos.

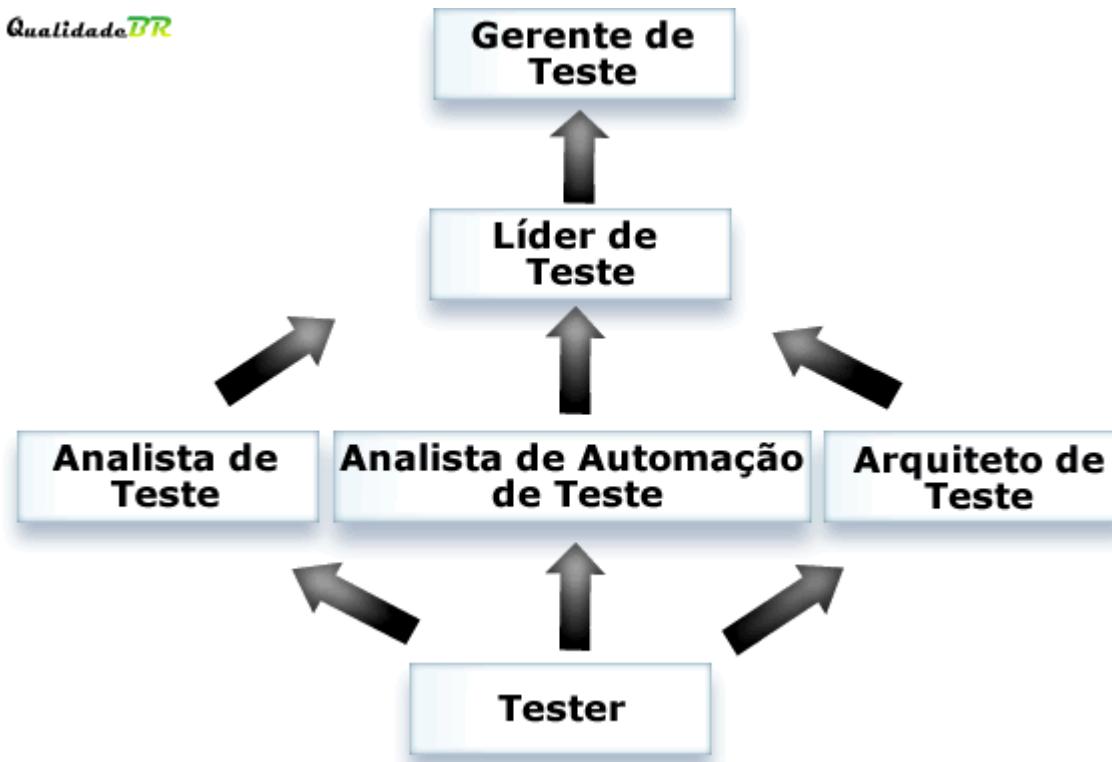


TESTE DE PROGRAMA

- Pode revelar a presença de defeitos, NÃO a ausência.
- É a principal técnica de validação para requisitos não funcionais, visto que o software é executado para ver como se comporta.
- Deve ser usado em conjunto com a verificação estática para fornecer cobertura completa de V&V.



ATÉ ONDE O TESTE PODE DEMANDAR?

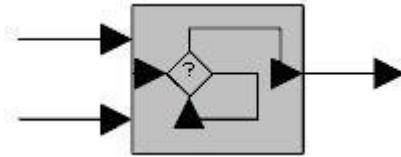




TESTE DE CAIXA-PRETA

- Testar todas as entradas e saídas desejadas.
- Não se está preocupado com o código, cada saída indesejada é visto como um erro.
- Não conhece como o sistema foi implementado.
- Teste não realizado pelo desenvolvedor.

TESTE CAIXA-BRANCA



- Feito tendo o conhecimento do código fonte.
- O objetivo é testar o código. Às vezes, existem partes do código que nunca foram testadas.
- Feito, normalmente, pela própria equipe técnica do software.
- São testados casos em que muitas vezes os usuários não conseguem testar.

TESTE DE UNIDADE

- Busca testar a menor unidade testável do software.
- É o teste cujo objetivo é um “pedaço do código”. Não o fluxo completo.
- Não busca testar a funcionalidade toda do sistema (teste funcional). Apenas um “pedaço”.
- Considerado como sendo um teste do tipo caixa-branca.
- Feito, na maioria das vezes, pelos próprios desenvolvedores (TDD – Mais voltado para boas práticas de desenvolvimento do que qualidade em si).
- Explorado principalmente pelos frameworks XUnit (JUnit, PHPUnit e NUnit).

TESTE DE INTEGRAÇÃO

- Garante que um ou mais componentes combinados (ou unidades) funcionam.
- Podemos dizer que um teste de integração é composto por diversos testes de unidade.

TESTE OPERACIONAL

- Garante que a aplicação pode rodar muito tempo sem falhar.
- São testes em cima da aplicação e do sistema operacional em conjunto, buscando garantir o funcionamento a longo prazo.

TESTE DE REGRESSÃO

- Busca testar toda a aplicação novamente, sempre que algo for alterado.
- Teste de regressão não corresponde a um nível de teste, mas é uma estratégia importante para redução de “efeitos colaterais”.
- Pode ser feito de forma manual ou automatizado.

TESTE FUNCIONAL OU DE SISTEMA

- Busca testar as funcionalidades do sistema.
- Construídos baseado nas regras de negócio presentes na documentação.
- Pode detectar erros na documentação.

TESTE DE INTERFACE OU USABILIDADE

- Verifica se a naveabilidade e os objetivos da tela funcionam como especificados e se atendem da melhor forma ao usuário.
- Voltado para garantir a usabilidade e a garantia de que todo o fluxo está completo no sistema.

TESTE DE PERFORMANCE

- Verifica se o tempo de resposta é o desejado para o momento de utilização da aplicação.
- Pode ser implementado utilizando várias ferramentas, inclusive o firebug.

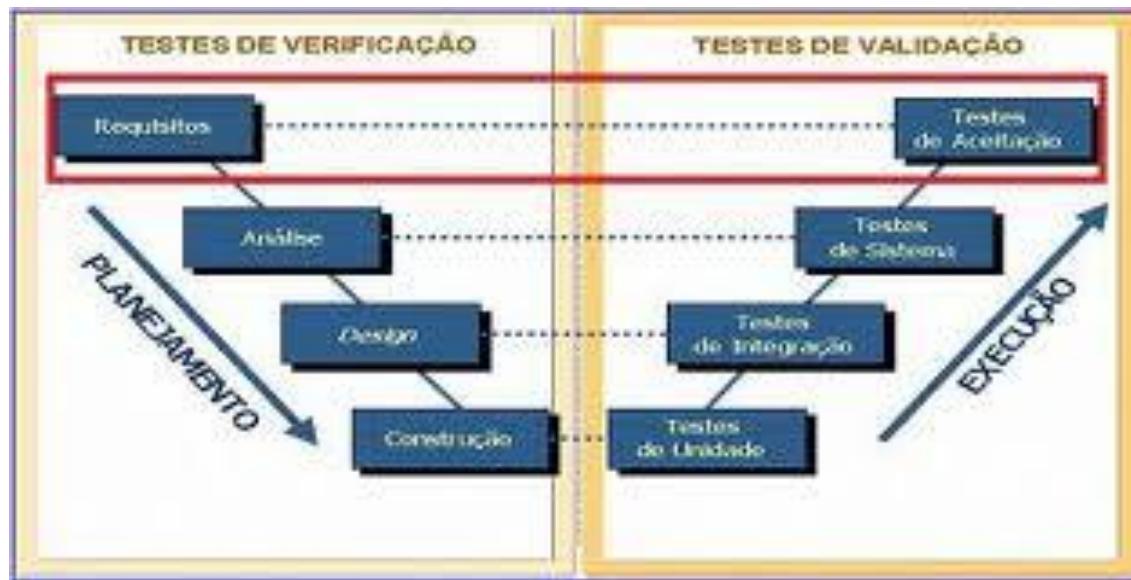
TESTE DE CARGA

- Verifica o funcionamento da aplicação com a utilização de uma quantidade grande de usuários simultâneos.
- Feito através de ferramentas que simulam essa carga (Exemplo: JMeter)

TESTE DE ESTRESSE

- É realizado para submeter o software a situações extremas.
- Testar os limites do software e avaliar seu comportamento.
- Busca avaliar até quando o software pode ser exigido e quais as falhas (se existirem) decorrentes do teste.

PLANEJAMENTO E EXECUÇÃO DOS TESTES

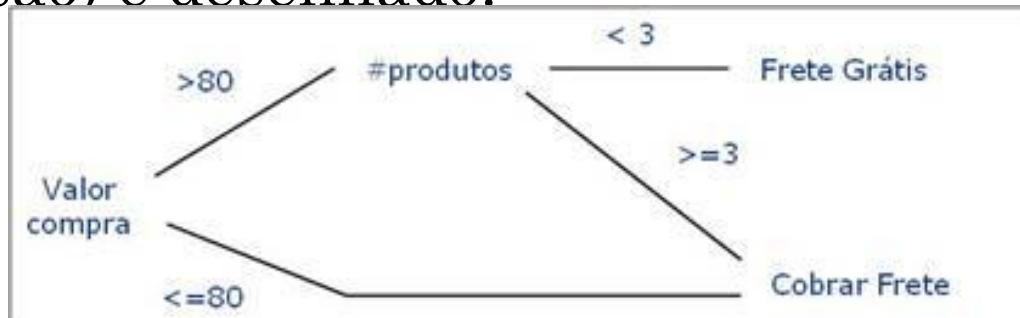


ANÁLISE DO VALOR LIMITE

- Por razões não completamente identificadas, um grande número de erros tende a ocorrer nos limites do domínio de entrada invés de no “centro”.
- Exemplo: se uma condição de entrada especifica uma faixa de valores limitada em a e b , casos de teste devem ser projetados com valores a e b e imediatamente acima e abaixo de a e b . Exemplo: Intervalo = {1..10}; Casos de Teste à {1, 10, 0, 11}.

GRAFO DE CAUSA-EFEITO

- Construção em 4 passos:
 1. Para cada módulo, **Causas** (condições de entrada) e **efeitos** (ações realizadas às diferentes condições de entrada) são relacionados, atribuindo-se um identificador para cada um.
 - Causa: valor da compra > 60 e $\#$ produtos < 3
 - Efeito: frete grátis
 2. Em seguida, um grafo de causa-efeito (árvore de decisão) é desenhado:



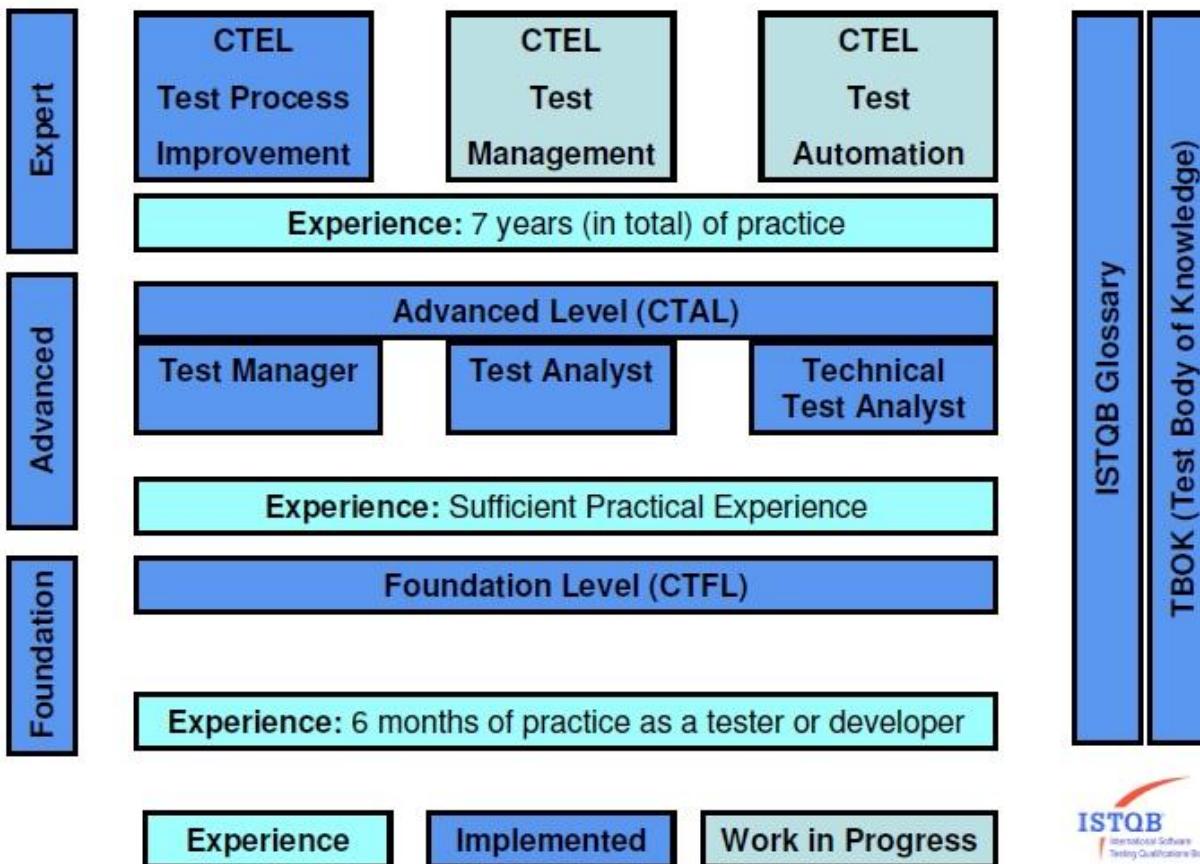
GRAFO DE CAUSA-EFEITO

3. Transforma-se o grafo numa tabela de decisão:

Causa	Valor da compra	> 60	> 60	<= 60
	#Produtos	< 3	>= 3	--
Efeito	Cobrar frete		✓	✓
	Frete grátis	✓		

4. As regras da tabela de decisão são convertidas em casos de teste.

CERTIFICAÇÕES



FRASES QUE ACOBERTAM BUGS

- *"Depois eu escrevo o plano de testes..."*
- *"Vamos deixar os testes para a próxima fase..."*
- *"Na minha máquina funcionou..."*
- *"Temos que entregar o produto na semana que vem..."*

DEVEMOS SEMPRE CORRIGIR TODOS OS BUGS?

- "É de conhecimento geral entre os analistas de software que nunca se elimina o último bug de um programa. Os bugs são aceitos como uma triste realidade. Esperamos eliminá-los todos, um por um, mas nunca conseguiremos nos livrar deles."
DeMarco, Tom , Editora Campus, 91
- “A melhor qualidade é aquela que o cliente pode pagar.”

TIPOS DE TESTES

○ Referências:

- <http://qualidadesoftware.org.br/>
- <http://softwarequalitycenter.blogspot.com/2010/02/implantacao-de-testes-de-software-com.html>
- <http://qualidade-de-software.blogspot.com/2010/01/teste-de-estresse.html>
- http://www.macoratti.net/tst_sw1.htm
- <http://www.plugmasters.com.br/sys/materias/647/1/Qualidade-de-Software---Tipos-de-testes-em-software,-parte-1>
- <http://testesdesoftware.blogspot.com/2009/10/ferramentas-de-testes-de-software.html>

MÉTRICAS DE SOFTWARE - MOTIVAÇÃO

Um dos objetivos básicos da Engenharia de Software é: *a transformação da criação de sistemas software de uma maneira artística, indisciplinada e pouco entendível para uma forma devidamente **controlada, quantificada e previsível***

“Métricas de Software” é um assunto discutido há mais de 20 anos na engenharia de software ... e no entanto não é verificada sua utilização, na prática, pela grande maioria dos projetos de construção de software

Pesquisas realizadas em empresas de software indicam que mais da metade de grandes projetos de software se deparam com algum tipo de atraso, excesso de custo ou prazo ou algum fracasso na execução quando implantado

Falta de controle dos projetos

MÉTRICAS DE SOFTWARE - MOTIVAÇÃO

- “Não se pode gerenciar o que não se pode medir”.

Tom De Marco

- “Se você não sabe para onde você quer ir, qualquer caminho você pode seguir. Se você não sabe onde você está, um mapa não vai ajudar!”.

Roger Pressman

O QUE SÃO MÉTRICAS DE SOFTWARE?

- Uma métrica é a medição de um atributo (propriedades ou características) de uma determinada entidade (produto, processo ou recursos). Exemplos:
 - Tamanho do produto de software (ex: Número de Linhas de código)
 - Número de pessoas necessárias para implementar um caso de uso
 - Número de defeitos encontrados por fase de desenvolvimento
 - Esforço para a realização de uma tarefa
 - Tempo para a realização de uma tarefa
 - Custo para a realização de uma tarefa
 - Grau de satisfação do cliente (ex: adequação do produto ao propósito, conformidade do produto com a especificação)

POR QUE MEDIR SOFTWARE?

- Entender e aperfeiçoar o processo de desenvolvimento
- Melhorar a gerência de projetos e o relacionamento com clientes
- Reduzir frustrações e pressões de cronograma
- Gerenciar contratos de software
- Indicar a qualidade de um produto de software
- Avaliar a produtividade do processo
- Avaliar os benefícios (em termos de produtividade e qualidade) de novos métodos e ferramentas de engenharia de software
- Avaliar retorno de investimento

PROPRIEDADES DESEJÁVEIS DE UMA MÉTRICA

- Facilmente calculada, entendida e testada
- Passível de estudos estatísticos
- Expressa em alguma unidade
- Obtida o mais cedo possível no ciclo de vida do software
- Repetível e independente do observador
- Sugere uma estratégia de melhoria

RESUMINDO...

- Uma métrica deve ser:
 - Válida: quantifica o que queremos medir
 - Confiável: produz os mesmos resultados dadas as mesmas condições
 - Prática: barata, fácil de computar e fácil de interpretar
- Dois contextos para medição de software
 - Processo: ex. produtividade
 - Produto: ex. qualidade

CATEGORIZAÇÃO DE MÉTRICAS

- Métricas orientadas a tamanho
 - São medidas diretas do tamanho dos artefatos de software associados ao processo por meio do qual o software é desenvolvido.
 - Ex.: esforço, custo, no. KLOC, no. páginas de documentação, no. erros
- Métricas orientadas por função
 - Consiste em um método para medição de software do ponto de vista do usuário, determinando de forma consistente o tamanho e a complexidade de um software.

CATEGORIZAÇÃO DE MÉTRICAS

○ Métricas de produtividade

- Concentram-se na saída do processo de engenharia de software.
- Ex.: no. de casos de uso/iteração.

○ Métricas de qualidade

- Oferecem uma indicação de quanto o software se adequa às exigências implícitas e explícitas do cliente.
- Ex.: erros/fase

○ Métricas técnicas

- Concentram-se nas características do software e não no processo por meio do qual o software foi desenvolvido.
- Ex.: complexidade lógica e grau de manutenibilidade

O PROCESSO DE MEDIÇÃO

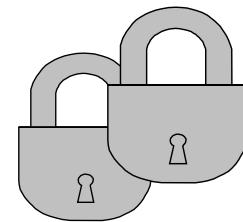
- É um processo cíclico que envolve:
 - Planejar
 - Medir
 - Analisar os dados
 - Tomar decisões baseadas na análise
 - Implementar as decisões
 - Voltar a planejar e medir

CARACTERÍSTICAS DE UM PROGRAMA EFETIVO DE MEDIÇÃO

- Escolha um conjunto adequado de métricas
- Relacione as métricas ao processo de tomada de decisão (suportado pela alta administração)
- Avalie processos e não pessoas (explique os objetivos da medição)
- Não use as métricas para punir
- Envolva várias pessoas na seleção e formulação das métricas
- Estabeleça alta prioridade (recursos, ferramentas, etc.)
- Integre o programa ao desenvolvimento de software
- Alinhe aos objetivos de negócio
- Padronize e documente
- Compartilhe as métricas obtidas
- Institucionalize como parte da cultura da organização
- Integre com o programa de melhorias (ilustre o progresso e as melhorias obtidos a partir do programa)
- Ofereça planos de ação

CUIDADO COM...

- Elaborar um política de controle de acesso
 - Apenas pessoas autorizadas devem ter acesso a certos tipos de dados
- Evitar o uso indevido dos dados
 - Avaliação de pessoas
 - Comparação entre projetos, grupos ou áreas da empresa de forma indevida
 - Publicação de informações que foram fornecidas de forma confidencial



Atenção: O uso indevido dos dados impacta fortemente e negativamente um programa de medições

POR QUE É TÃO DIFÍCIL ESTIMAR?

- É difícil conhecer se é possível desenvolver o produto desejado pelo cliente antes de conhecer os detalhes do projeto.



POR QUE É TÃO DIFÍCIL ESTIMAR?

- Desenvolvimento é um processo gradual de refinamento
 - Incerteza da natureza do produto contribui para a incerteza da estimativa
 - Requisitos e escopo mudam
 - Defeitos são encontrados e demandam retrabalho
 - Produtividade varia

O PROCESSO DE ESTIMATIVAS

1. Estimar o tamanho do produto
2. Estimar o esforço
3. Estimar o prazo
4. Fornecer estimativas dentro de uma faixa permitida e refinar essa faixa à medida que o projeto progride

PRINCIPAIS BARREIRAS



Falta de comprometimento da alta gerência

Medir custa caro

Os maiores benefícios vêm a longo prazo

Má utilização das métricas

Grande mudança cultural necessária

Dificuldade de estabelecer medições apropriadas e úteis

Interpretações dos dados realizadas de forma incorreta

Obter o comprometimento de todos os envolvidos e impactados

Estabelecer um programa de medições é fácil, o difícil é manter!!

MAS PODEMOS CONTORNAR...

Foco desde os estágios iniciais da melhoria de processo

Medição faz parte do TODO

Começar Pequeno

Selecionar um conjunto coerente

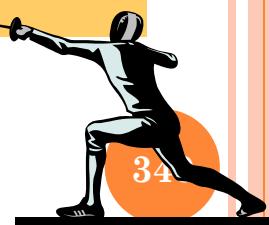
É importante definir cada detalhe da métrica

Descartar o que não estiver sendo útil

Fornecer as informações corretas, para as pessoas certas

“Aregar valor”, ao invés de gerar apenas dados

Criado por Paulo Henrique Ladeira



MAS PODEMOS CONTORNAR...

Incentivar a equipe de desenvolvimento a fazer uso das métricas

Envolvimento de todos os impactados

Estabelecer as expectativas

Educação e Treinamento

Ganhar Confiança

Adotar uma Abordagem Evolucionária

Compreender que a Adoção leva Tempo

Criado por Paulo Henrique Ladeira



MÉTRICAS FONTE

- http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CEAQFjAB&url=http%3A%2F%2Fwww.cin.ufpe.br%2F~if720%2Fsli des%2Fintroducao-a-metricas-de-software.ppt&ei=IquyUOzbGIfm8gTT_4C4CA&usg=AFQjCNGUjFVr1HkROf8U0n-3xaaLHCg7Gg&sig2=unvm5KDqxSs_q-xfgEiiqQ&cad=rja