CURSO DE SISTEMAS DE INFORMAÇÃO

QUALIDADE DE SOFTWARE

Prof. André Castro

Ementa

1. Introdução
1.1 O que é Qualidade de Software
1.2 Qualidade de Produto de Software
1.3 Qualidade do Processo de Software
1.4 Qualidade x Produtividade
1.5 Normas e Organismos Normativos

2. Qualidade de Processo de Software
2.1 Definição de Processos de Software
2.2 Modelos de Ciclo de Vida (Modelos de Processo)
2.3 Normas e Modelos de Apoio à Definição de Processos de Software
2.3.1 Série ISO 9000
2.3.2 ISO/IEC 12207
2.3.3 CMMI
2.3.4 ISO/IEC 15504
2.3.5 MR MPS.BR

3. Qualidade de Produto de Software
3.1 Modelo de Qualidade de Produto
3.2 Normas de Qualidade de Produto
3.2.1 ISO/IEC 9126
3.2.2 ISO/IEC 14598
3.2.3 ISO/IEC 12119

4. Medição de Software
4.1 O Processo de Medição de Software
4.2 GQM
4.3 Definição de Medidas de Software
4.4 O Plano de Medição

5. Verificação e Validação
5.1 Revisões Técnicas e Inspeções
5.2 Estratégias e Técnicas de Testes
5.3 Processo de Teste
5.4 Tipos de Testes

6. Métodos Ágeis e a Qualidade de Software
6.1 Métodos Ágeis
6.2 SCRUM
6.3 Qualidade de Software nos Métodos Ágeis

GQM

GOAL, QUESTION, METRIC

Medição de Software

GQM – Goal, Question, Metric

Fonte: Basili, V.R.; Caldiera, G.; Rombach, H.D.; The Goal Question Metric Approach.

Justificativa para medição

– ver onde estão os gargalos do processo de desenvolvimento

– identificar as características boas (desejáveis) de uma classe de artefatos

– identificar as características ruins (a serem evitadas) de uma classe de artefatos

– identificar riscos reais

– auxiliar na construção de conjuntos de padrões, diretrizes e ferramentas efetivamente úteis

– avaliar o efeito das mudanças de tecnologia

Conceitos Básicos

A idéia básica de GQM é derivar **métricas** de software a partir de perguntas e objetivos.

Este método foi originalmente criado por Victor Basili e Weis como resultado de experiências práticas e pesquisas acadêmicas.

Goal

Quais são as metas/objetivos?

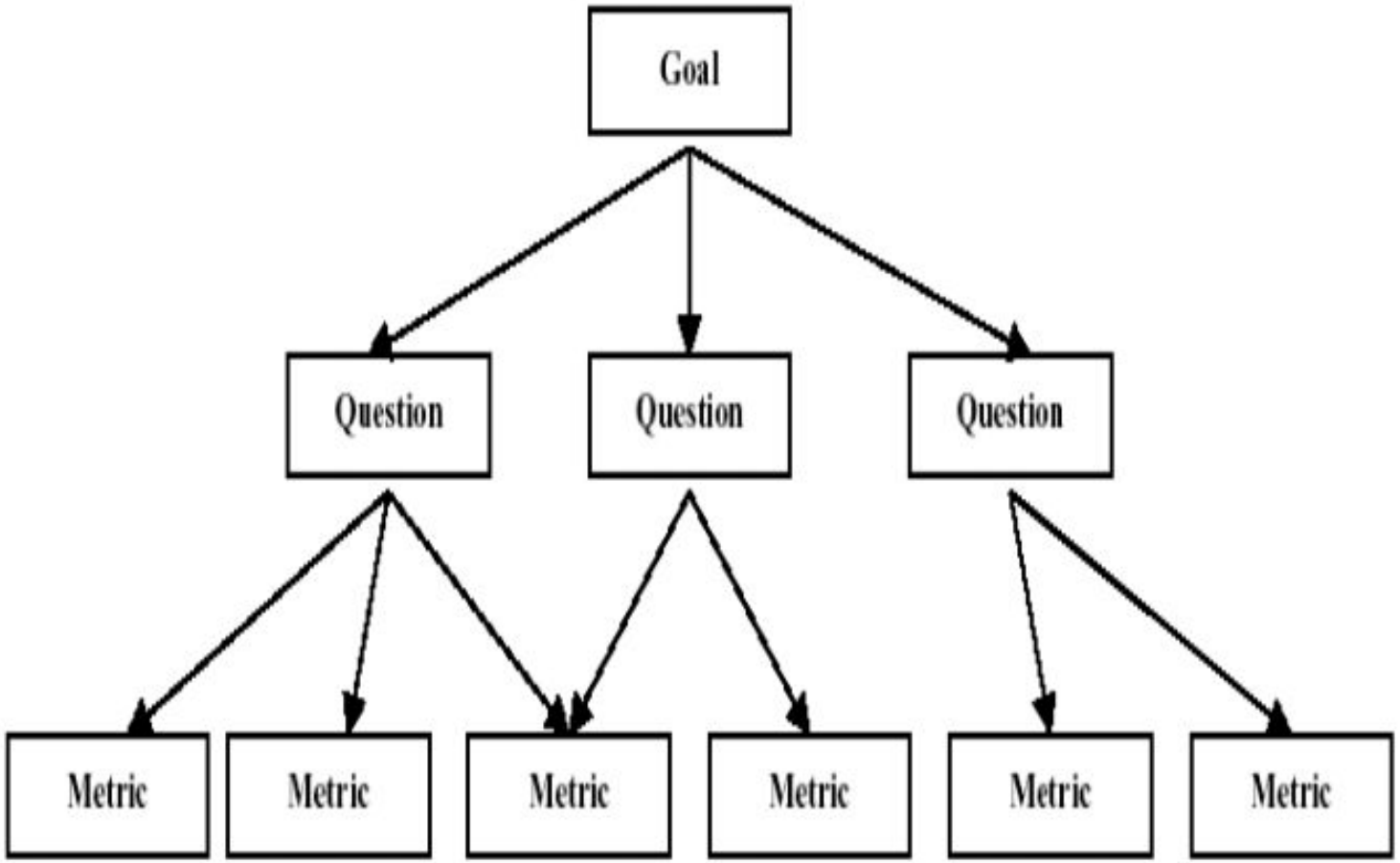
Question

Quais questões se deseja responder?

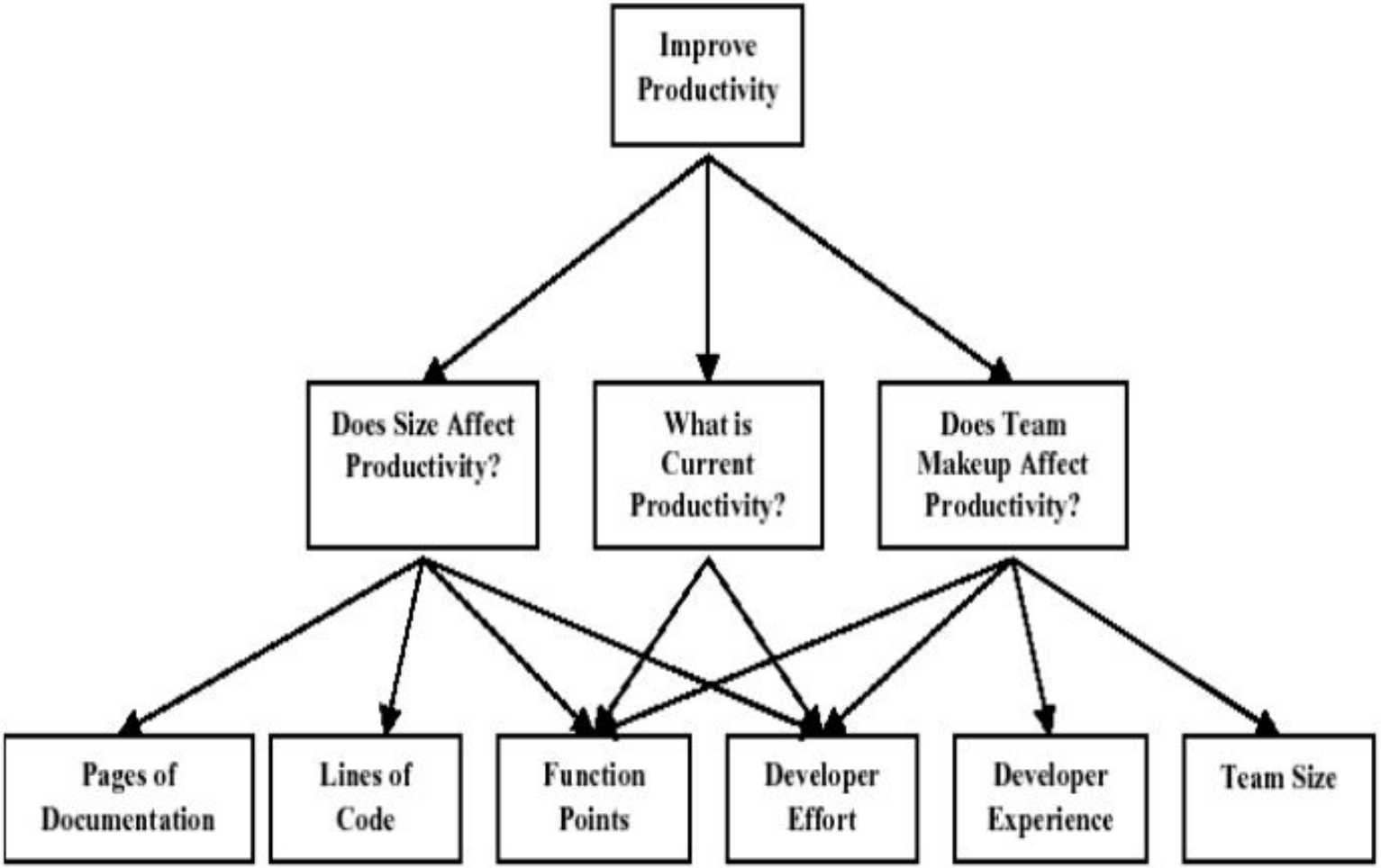
Metric

Quais métricas poderão ajudar?

Idéia Básica do GQM



Exemplo


Improve Productivity
Does Size Affect Productivity?
What is Current Productivity?
Does Team Makeup Affect Productivity?
Pages of Documentation
Lines of Code
Function Points
Developer Effort
Developer Experience
Team Size

Exemplo

Table 1. The GQM Approach to Extract Information Related to the Factor Dynamism

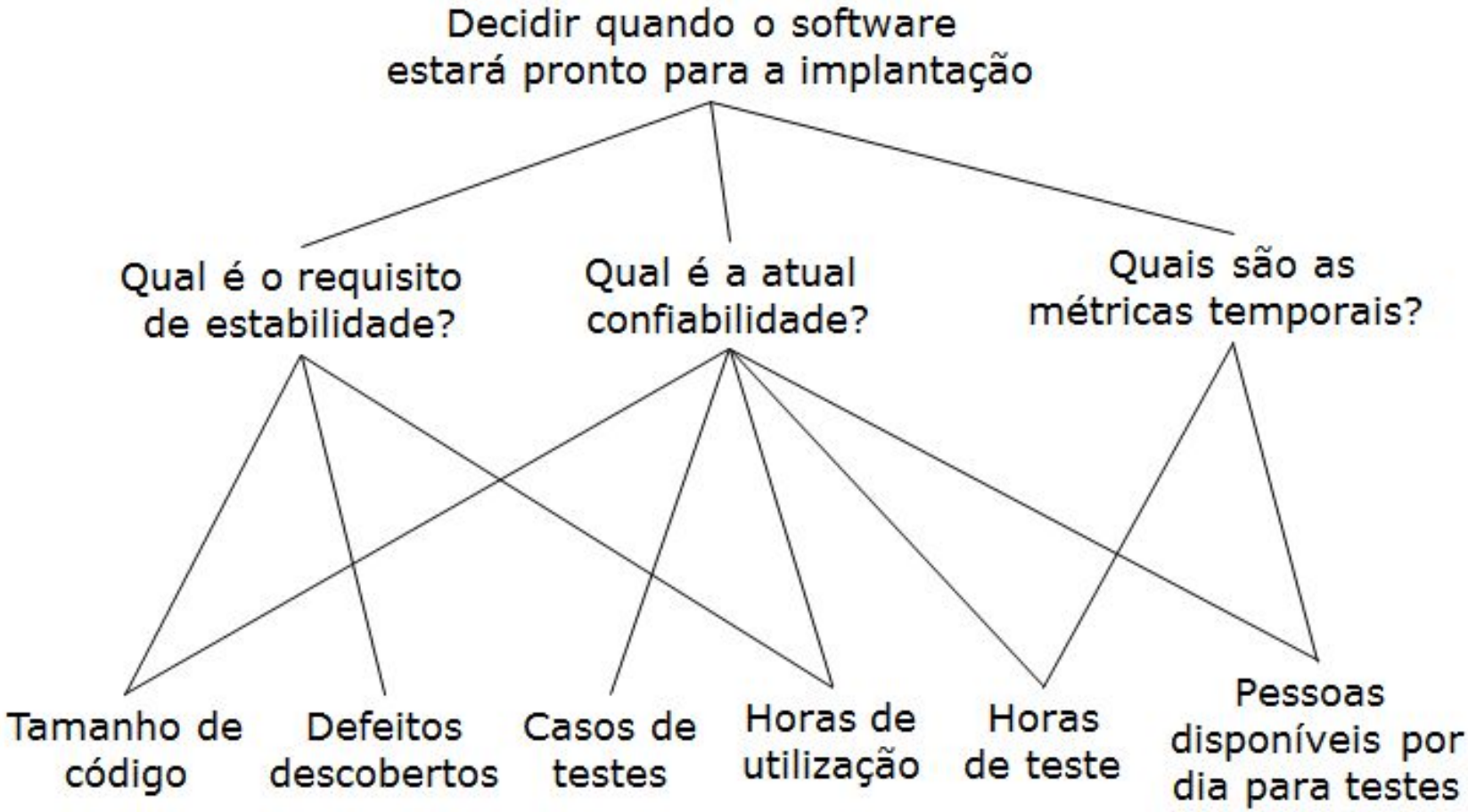
Purpose:	Estimate.
Issue:	Frequency of software requirements changes caused by the problem context changes.
Object:	Software requirements.
Viewpoint:	Process manager.

Question:	Which is the main reason for requirements changes in the system?
Metric(s):	- % of requirements changes as a consequence of the problem context changes. - % of requirements changes as a consequence of the visualization of delivered parts of system (prototyping). - % of requirements changes as a consequence of the incorrect specification of requirements.

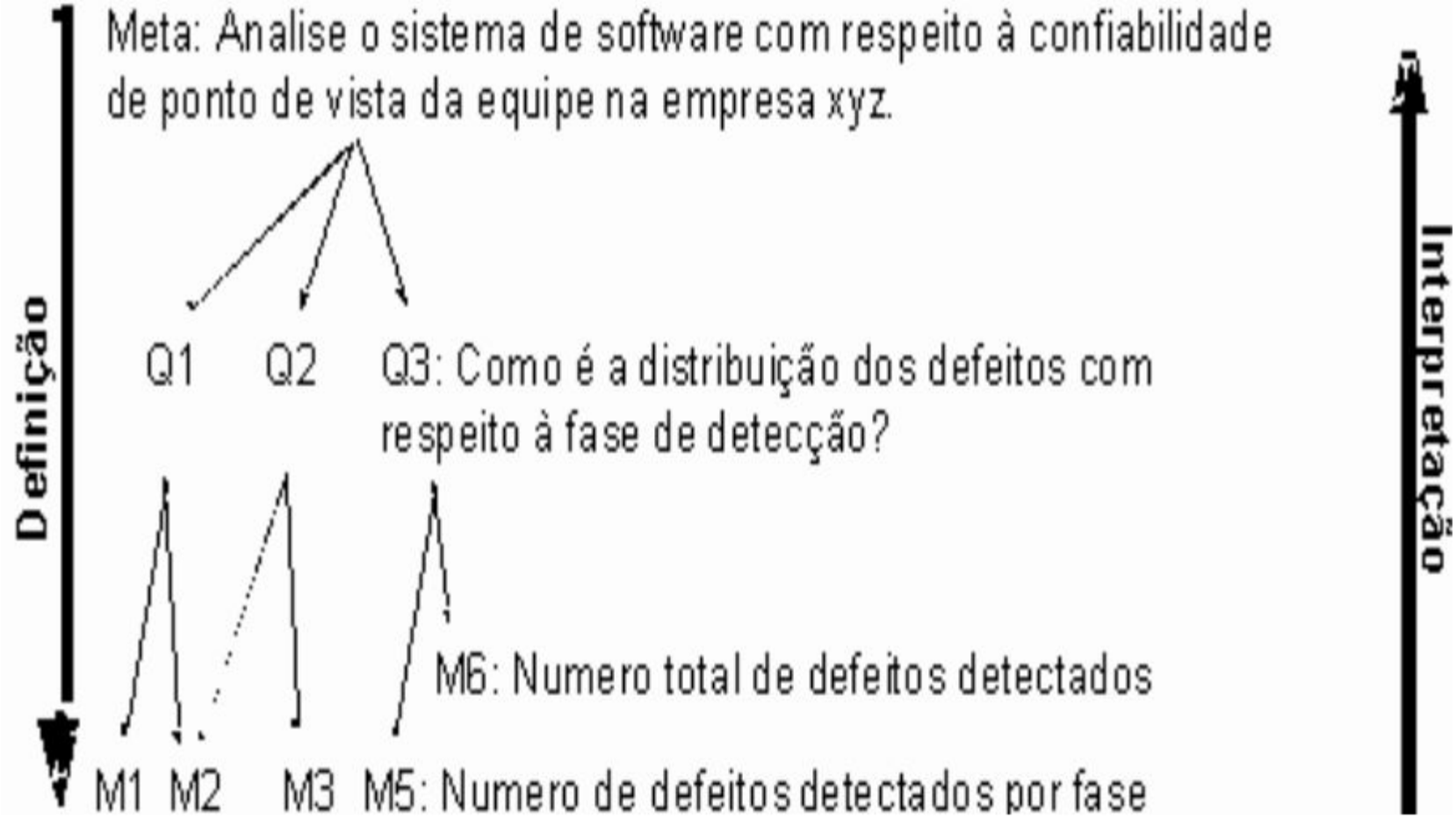
Question:	Which is the frequency of requirements changes in cases that these changes occur as a consequence of problem context changes?
Metric(s):	- % of requirements changes / month (from the total established until now).

Fonte: SOARES, L. S., Braga, J. L., LEAL, A. L. C., SILVA, C. H. O., CAMPOS, J. P.
Risk profile assessment for software development teams: a first step towards best practices adoption. In: CLEI 2008 - Conferencia Latinoamericana de Informática. Santa Fé, Argentina: SADIO- Sociedad Argentina de Informática, 2008, Argentina. Anais do XXXIV Clei.. Argentina: , 2008. v.XXIV. p.509 - 518

Exemplo


Decidir quando o software estará pronto para a implantação
Qual é o requisito de estabilidade?
Qual é a atual confiabilidade?
Quais são as métricas temporais?
Tamanho de código
Defeitos descobertos
Casos de testes
Horas de utilização
Horas de teste
Pessoas disponíveis por dia para testes

Componentes



Níveis

- Conceitual
  - Definir o objetivo para o objeto a ser medido;
- Operacional
  - Conjunto de questões;
- Quantitativo
  - Dados a serem apurados ou medidos;

Etapas

- Desenvolvimento do Plano GQM
  - Pré-estudo;
  - Elaboração do Plano GQM;
  - Elaboração do Plano de Avaliação
- Execução do Plano de Avaliação
  - Coleta de Dados;
  - Tratamento dos Dados;
- Preparação dos Resultados
  - Preparação da Documentação Final;
  - Composição da Base de Experiências;

Execução do Plano de Avaliação

- Coleta dos dados
  - A partir de formulários projetados na etapa de desenvolvimento do GQM;
  - Validação dos dados coletados de acordo com o objetivo do plano GQM;

- Tratamento dos dados
  - Análise e interpretação dos dados coletados;
  - Técnicas de levantamento e estatística;

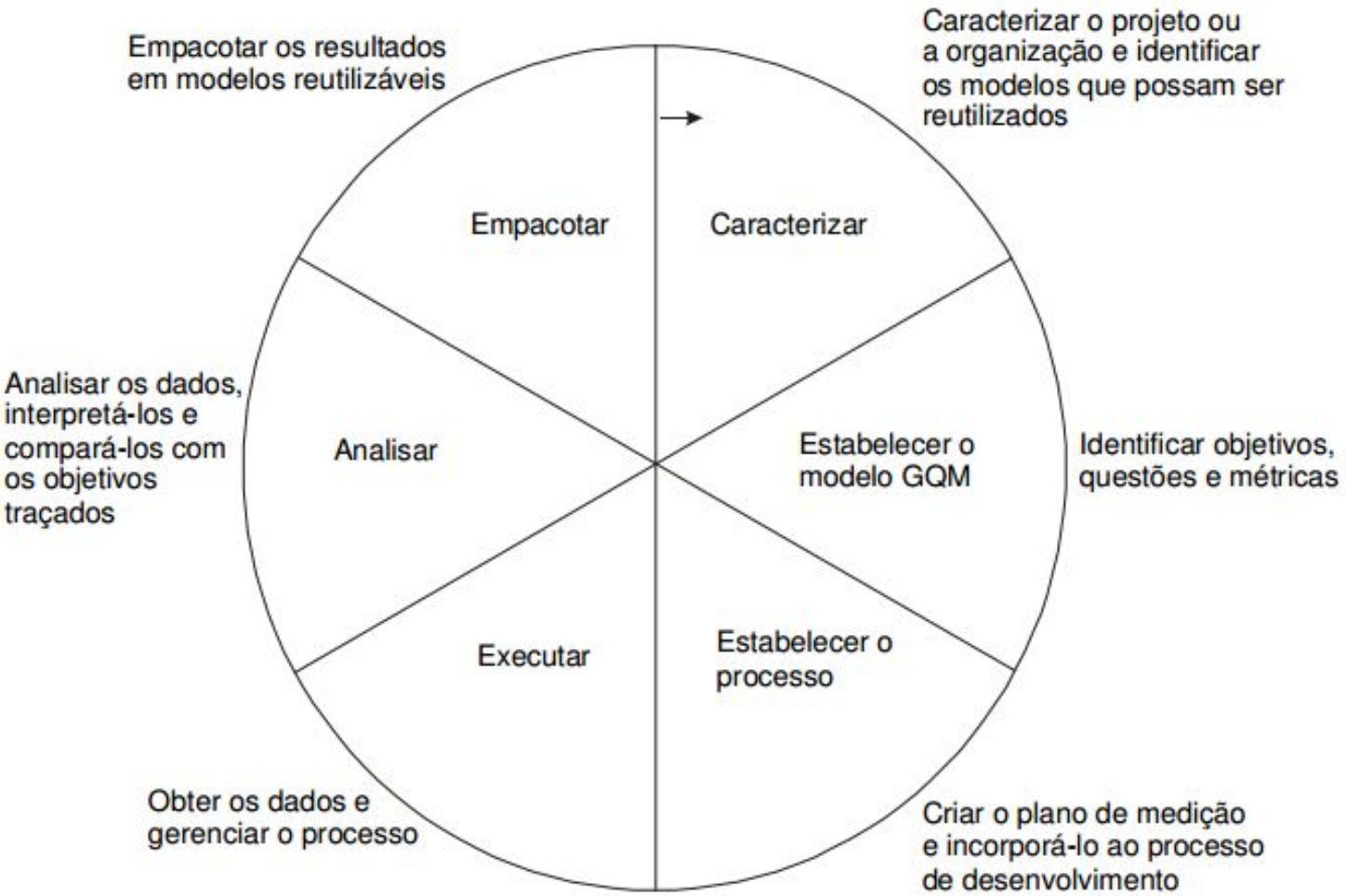
Preparação da documentação final
  - Reúne-se todo o material gerado ao longo das etapas do plano GQM;

- Composição da base de experiências
  - Reutilização da base de conhecimento;
  - Proposta de estratégias de melhoria da qualidade;

Alguns Softwares

- GQM-Plan
- GQMaspect
- GQM DIVA
- TEAM
- MetriFlame

Processo GQM



Identificação dos Objetivos

Object *What will be analyzed?*	**Analyse**	processes, products, resources
Purpose *Why will the object be analyzed?*	**for the purpose of**	characterization, evaluation, monitoring, prediction, control, improvement
Quality Focus *What property of the object will be analyzed?*	**with respect to**	cost, correctness, defects, changes, reliability, user friend-liness, etc.
Viewpoint *Who will use the data collected?*	**from the viewpoint of**	user, senior manager, project manager, developer, etc.
Environment *In which environment does the analysis take place?*	**in the following con-text**	organization, project, problem, processes, etc.

Exemplo

Goal	Purpose Issue Object (process) Viewpoint	Improve the timeliness of change request processing from the project manager's viewpoint
Question		What is the current change request processing speed?
Metrics		Average cycle time Standard deviation % cases outside of the upper limit
Question		Is the performance of the process improving?
Metrics		$\frac{\text{Current average cycle time}}{\text{Baseline average cycle time}} * 100$ Subjective rating of manager's satisfaction

Exemplo

Goal	Purpose Issue Object (process) Viewpoint	Improve the timeliness of change request processing from the project manager's viewpoint
Question	Q1	What is the current change request processing speed?
Metrics	M1 M2 M3	Average cycle time Standard deviation % cases outside of the upper limit
Question	Q2	Is the (documented) change request process actually performed?
Metrics	M4 M5	Subjective rating by the project manager % of exceptions identified during reviews
Question	Q3	What is the deviation of the actual change request processing time from the estimated one?
Metrics	M6 M7	$\frac{\text{Current average cycle time} - \text{Estimated average cycle time}}{\text{Current average cycle time}} * 100$ Subjective evaluation by the project manager
Question	Q4	Is the performance of the process improving?
Metrics	M8	$\frac{\text{Current average cycle time}}{\text{Baseline average cycle time}} * 100$
Question	Q5	Is the current performance satisfactory from the viewpoint of the project manager?
Metrics	M7	Subjective evaluation by the project manager
Question	Q6	Is the performance visibly improving?
Metrics	M8	$\frac{\text{Current average cycle time}}{\text{Baseline average cycle time}} * 100$

Identificação dos Objetivos

• Analisar o processo de desenvolvimento com o objetivo de entender as causas de perda de confiabilidade do ponto de vista da equipe de desenvolvimento e no contexto do projeto XYZ.

• Analisar métodos com o objetivo de determinar o efeito da complexidade sobre o número de defeitos encontrados do ponto de vista da equipe de desenvolvimento e no contexto de programas C.

Identificação dos Objetivos

• Analisar <objeto do estudo> com o propósito de <propósito> com relação ao seu (à sua) <foco> do ponto de vista de <perspectiva> e no contexto <contexto>.

Objeto	Propósito	Foco	Perspectiva	Contexto
produto	caracterizar	eficácia	desenvolvedor	pessoas
processo	monitorar	custo	mantenedor	objetos
modelo	avaliar	confiabilidade	gerente	linguagens
métrica	predizer	manutenibilidade	direção	tecnologia
teoria	controlar	portatilidade	cliente	projeto x
técnica	modificar	...idade	usuário	
			pesquisador	

Fatores de Sucesso

Técnicos

– Quais as métricas utilizadas

– Procedimentos de coleta de dados
- não intrusivo
- disponibilidade de ferramentas de coleta e análise
- independente de pessoas

– Qualidade das medidas
- rigor da medição
- rigor da análise

– Treinamento

– Comunicação dos resultados
- feedback
- observação da utilidade dos resultados pelos desenvolvedores e gerentes

Fatores de Sucesso

Organizacionais

– Efetivo compromisso das pessoas envolvidas com o programa de medição

– Disponibilidade de suficientes recursos

– Apoio gerencial

– Maturidade da organização

– Crenças da organização

Aspectos Positivos do GQM

Aplicabilidade generalizada

– Não existem restrições quanto aos objetivos
  - desde caracterização de processos ou técnicas
  - até controle e melhoria

Identificação e ajuste das métricas

– as métricas são selecionadas com vistas aos objetivos

– podem ser adaptadas às características dos projetos ou das organizações

Aspectos Positivos

- Interpretação das medidas
  - o modelo objetivo, questão e métrica também estabelece uma diretriz para a interpretação das medidas obtidas

- Apoio metodológico
  - GQM descreve um processo para o planejamento e a execução de um programa de medição

- Envolvimento de todos os interessados
  - o modelo visa atingir objetivos explícitos e de interesse dos envolvidos

- Separação de Papéis

Aspectos a Considerar

- Proteção dos dados sensitivos
  - a necessidade e o uso dos dados é definido e acordado a priori
  - a estrutura pode ser utilizada para criar mecanismos de controle de acesso
- Aspectos éticos
  - o processo de medição, análise e divulgação é estabelecido a priori permitindo os envolvidos a se manifestarem quanto a possíveis invasões de privacidade ou divulgação de dados confidenciais

Críticas ao GQM

- GQM é útil para identificar os objetivos da medição
- Não se preocupa com os problemas relacionados com a medição em si
  - Viabilidade
  - Economicidade
- quais os benefícios esperados
- quanto custa (custará) medir, armazenar e processar?
  - Corretude do modelo de medição
  - Corretude e confiabilidade dos modelos estatísticos
  - Completeza (se não foram esquecidas varáveis relevantes)
  - Técnicas de medição
  - Escalas das medidas
  - Responsabilidade pela análise dos resultados
  - . . .

V & V

Verificação e Validação de Software

V & V

Objetivo: assegurar que o software que o software
– cumpra as suas especificações e
– atenda às necessidades dos usuários e clientes.

• Verificação: IEEE 1012
– “Estamos construindo certo o produto?”
– O software deve está de acordo com a sua especificação.

• Validação: IEEE 1012
– “Estamos construindo o produto certo?”
– O software deve atender às necessidades dos usuários.

• Ocorrem em todo o ciclo de vida completo
– Revisões de requisitos, revisões de design, testes de código

Fonte: http://www.dimap.ufrn.br/~jair/ES/slides/VerificacaoValidacao.pdf

Diferenças entre V & V

A Verificação é uma atividade, a qual envolve a análise de um sistema para certificar se este atende aos requisitos funcionais e não funcionais.

A Validação, é a certificação de que o sistema atende as necessidades e expectativas do cliente. O processo de Validação e Verificação, não são processos separados e independentes.

Leia mais em: A Importância da validação e da verificaçãoLeia mais em: A Importância da validação e da verificação http://www.devmedia.com.br/a-importancia-da-validacao-e-da-verificacao/24559#ixzz2PsnIPoDg

O que deve ser verificado

- Fatores de Qualidade Operacionais
  - Correção
  - Eficiência ou desempenho
  - Robustez
  - Confiabilidade
  - Usabilidade
  - Utilidade e validade

- Fatores de Qualidade de Revisão
  - relacionados com a manutenção, evolução e avaliação do software

- Fatores de Qualidade de Transição
  - relacionados com a instalação, reutilização e interação com outros produtos

O que deve ser verificado - Conceitos

1) Falta (Fault): É inserida no software quando um desenvolvedor comete algum equívoco. Um equívoco pode causar várias faltas ao mesmo tempo que vários enganos pode causar uma falta idêntica.

2) Falha (Failure): Representa um comportamento incorreto apresentado por um software em conseqüência de uma falta.

3) Erro (Error): Representa o quanto um resultado é incorreto.

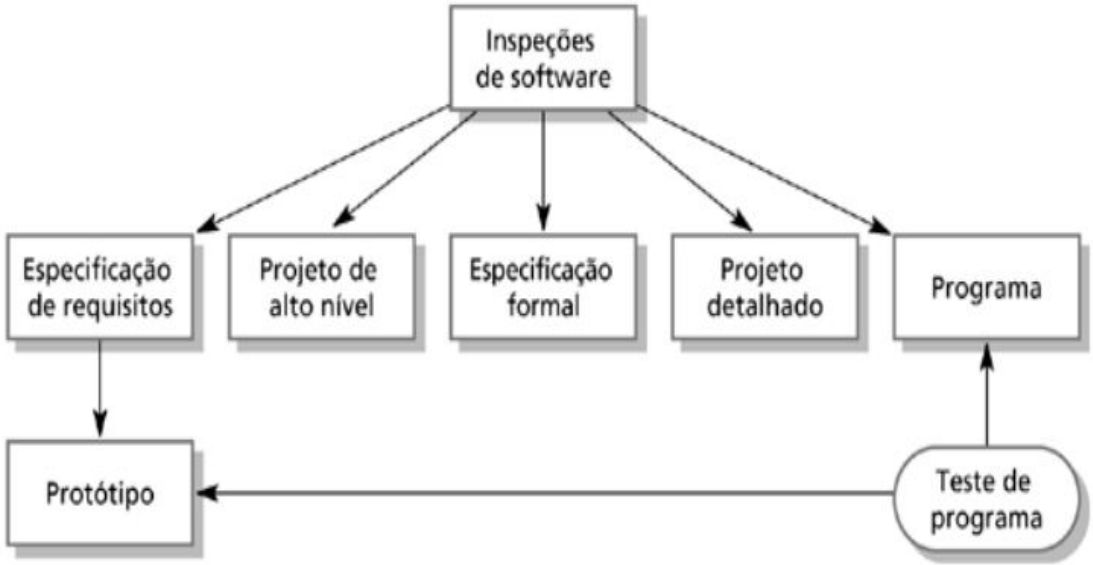
4) Defeito (Defect): Termo genérico para falta, falha ou erro..

Técnicas de V & V

- Inspeções de software (V & V estática)
  - Análise da documentação e código fonte do software
  - Pode ser auxiliado por ferramentas de depuração
- Testes de software (V & V dinâmica)
  - O programa ou um protótipo devem ser executados
  - Casos de testes deve ser elaborados: dados de entrada e comportamento esperado.

Figura 22.1

Verificação e validação dinâmica e estática.

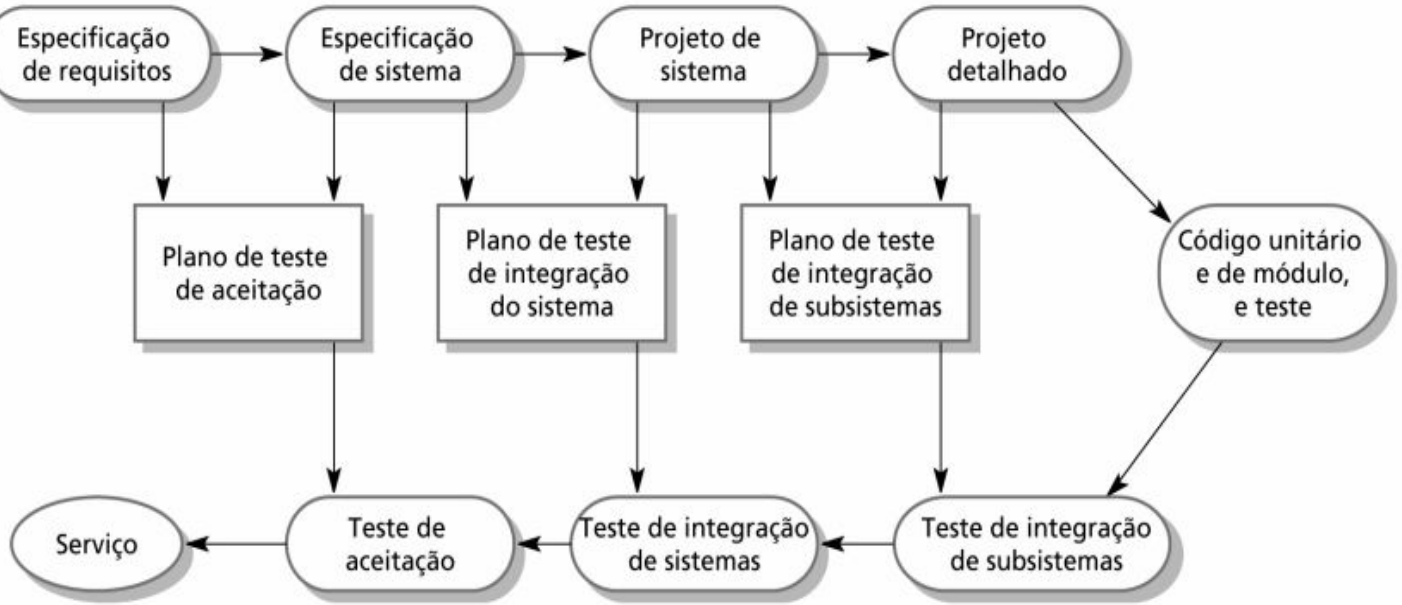


Livro Sommerville

Plano para V & V

- O processo de V&V ocorre durante todo o ciclo de vida
- Precisa ser planejado em conjunto com outras atividades do processo de software

Figura 22.3 Plano de teste como ligação entre o desenvolvimento e os testes.



Livro Sommerville

Inspeções em V & V

- Características
  - –Técnica preventiva – permite a V & V antes do software ser codificado
  - –Mais barata
  - –Baseada na experiência do inspetor
  - –Mais aplicada a fatores de revisão e transição
  - –Pouco eficaz para fatores operacionais

- Aplicações mais comuns
  - –Inspeção de programa fonte (estática e dinâmica)
  - –Inspeção de documentos e modelos
  - –Desenvolvimento Cleanroom (*www.estig.ipbeja.pt/~eides/**CleanRoom**.ppt*)

Abordagens para verificação da correção

• Inspeções analíticas
  – Estática

• Rastreamento do código fonte (walkthrough) – percorre-se o código executando-o mentalmente.
  – Automatizada (dinâmica)

• Depuração – execução passo-a-passo e visualização de variáveis do programa

• Testes de correção de programas
  – Testes de unidade
  – Testes de integração

• Prova Formal de Programas
  – Utiliza métodos formais de desenvolvimento de software – técnicas de especificação, transformação e prova forma

Testes de Software

Elaboração de casos de testes baseados na especificação funcional
- Dados de entradas
- Comportamento esperado

Podem ser classificados
- Quanto ao objetivo
- Quanto ao escopo
- Quanto ao método

Aplicações em fatores operacionais
- Correção
- Usabilidade
- Desempenho
- Robustez

Tipos de teste quanto ao objetivo

Quanto ao Objetivo

• Testes de Defeitos
– Tem por objetivo encontrar defeitos – inconsistências entre o programa e a sua especificação.
– Verifica a correção – conhecido também por testes de correção
– Normalmente realizados com protótipos funcionais

• Testes de Validação
– Utilizado para demonstrar ao desenvolvedor e ao cliente do sistema que o software atende aos seus requisitos.
– Um teste bem sucedido visa mostrar que o sistema opera conforme especificado pelo cliente.

Quanto ao Método

• Testes Controlados
– Realizados em laboratório ou em condições operacionais sob a supervisão de um testador
– Baseados na geração de casos de testes
– Podem ser realizados com protótipos funcionais

Tipos de teste quanto ao objetivo

Quanto ao Método

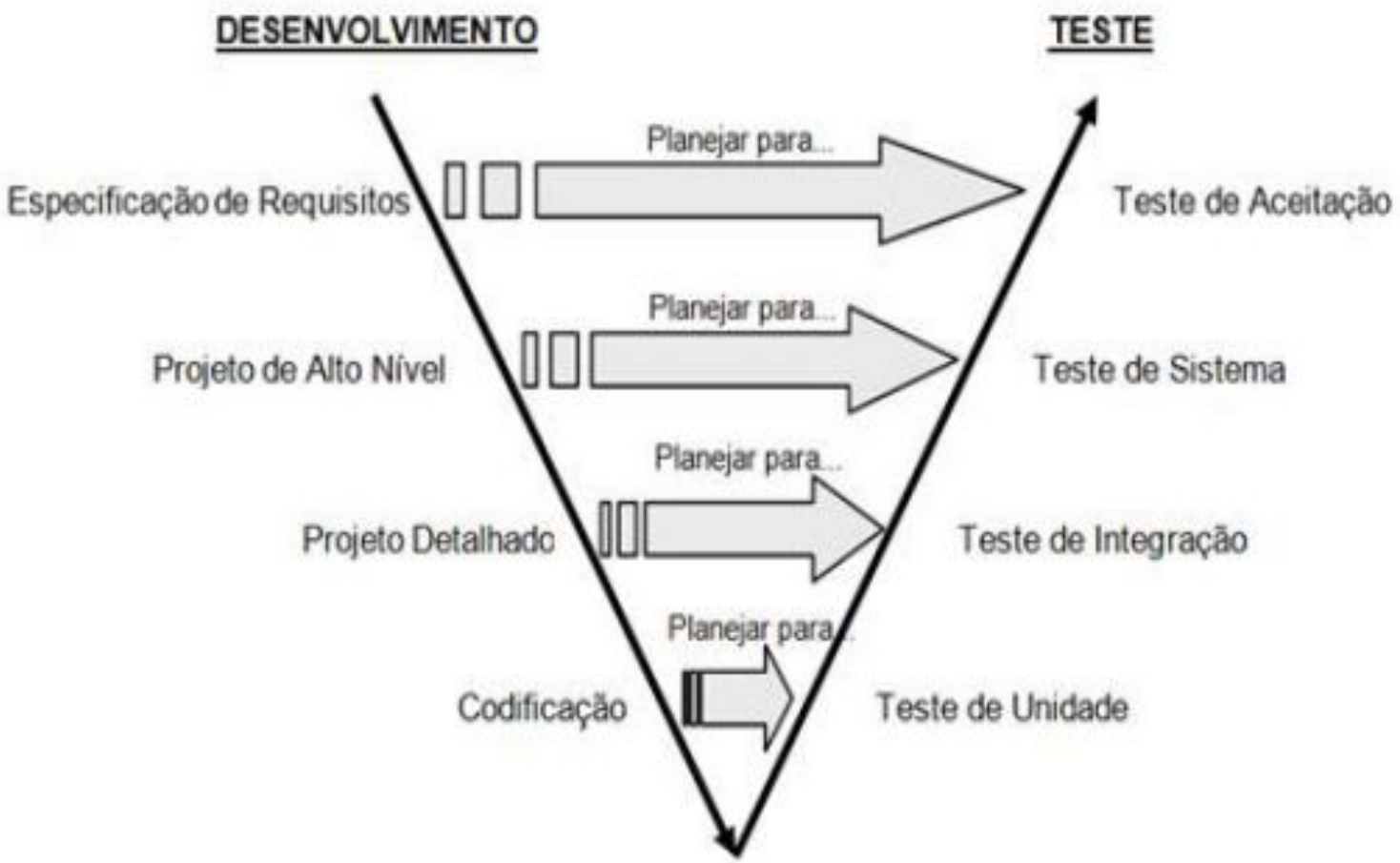
• Testes Estatísticos
– Utilizados para verificar como o software nas condições operacionais (versão beta)
– Avalia desempenho, robustez, confiabilidade, etc
– Utiliza programa de monitoramento e "logs" com ocorrências operacionais.
– Exemplos de medições:
  • Número de falhas observadas
  • Tempos de resposta
  • Tempos de execução

Quanto ao Escopo

• Testes de Unidade (componente)
– Conhecidos como testes em ponto pequeno
– São testadas as unidades individuais – funções, objetos, componentes.

• Testes de Integração (sistema)
– Conhecidos como testes em ponto grande
– Os componentes são integrados e o conjunto maior é testado – módulo e sub-sistemas.

Estratégicas de Testes



Verificação no CMMI

O.E. 1 - Preparar Verificação
Objectivo: É conduzida a preparação da verificação.
P.E. 1.1 - Seleccionar Produtos de Trabalho para a Verificação
Descrição: Seleccionar os produtos de trabalho a serem verificados e os métodos de verificação a serem utilizados para cada.
P.E. 1.2 - Estabelecer o Ambiente de Verificação
Descrição: Estabelecer e manter o ambiente necessário para suportar a verificação.
P.E. 1.3 - Estabelecer Procedimentos e Critérios de Verificação
Descrição: Estabelecer e manter procedimentos e critérios de verificação para os produtos de trabalho seleccionados.
O.E. 2 - Realizar Revisões Técnicas (peer reviews)
Objectivo: São realizadas revisões técnicas aos produtos de trabalho seleccionados.
P.E. 2.1 - Preparar as Revisões Técnicas
Descrição: Preparar as revisões técnicas dos produtos de trabalho seleccionados.
P.E. 2.2 - Realizar Revisões Técnicas
Descrição: Conduzir revisões técnicas aos produtos de trabalho seleccionados e identificar ocorrências resultantes da revisão técnica. P.E. 2.3 - Analisar Dados das Revisões Técnicas
Descrição: Analisar os dados relativos à preparação, realização e resultados das revisões técnicas.
O.E. 3 - Verificar os Produtos de Trabalho Seleccionados
Objectivo: Os produtos de trabalho são verificados face aos seus requisitos especificados.
P.E. 3.1 - Executar Verificação
Descrição: Executar a verificação dos produtos de trabalho seleccionados.
P.E. 3.2 - Analisar Resultados das Verificações e Identificar Acções Correctivas.
Descrição: Analisar os resultados de todas as actividades de verificação e identificar acções correctivas.

Validação no CMMI

O.E. 1 - Preparar Validação
Objectivo: É conduzida a preparação da validação.
P.E. 1.1 - Seleccionar os Produtos para a Validação
Descrição: Seleccionar os produtos, os componentes do produto a serem validados e quais os métodos de validação a serem utilizados para cada um.
P.E. 1.2 - Estabelecer Ambiente de Validação
Descrição: Estabelecer e manter o ambiente necessário para suportar a validação.
P.E. 1.3 - Estabelecer Procedimentos e Critérios de Validação
Descrição: Estabelecer e manter procedimentos e critério de validação.
O.E. 2 - Validar Produto ou Componentes do Produto
Objectivo: O produto ou componentes do produto são validados, de modo a, assegurar que estes são apropriados para utilização no ambiente de operação pretendido.
P.E. 2.1 - Realizar Validação
Descrição: Realizar a validação dos produtos ou componentes do produto seleccionados.
P.E. 2.2 - Analisar Resultados da Validação
Descrição: Analisar os resultados das actividades de validação e identificar ocorrências.

Métodos Ágeis

e

Qualidade de Software

Surgimento

Em 2001 Kent Beck, e mais 16 reconhecidos desenvolvedores, se reuniram para estipularem o "***Manifesto for Agile Software Development***" (Manifesto para Desenvolvimento Ágil de Software).

Ideias Básicas

"*Estamos descobrindo maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:*

- Indivíduos e interação entre eles mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano."

Princípios Básicos

- Simplicidade acima de tudo;
- Rápida adaptação incremental às mudanças;
- Desenvolvimento do software preza pela excelência técnica;
- Projetos de sucesso surgem através de indivíduos motivados, e com uma relação de confiança entre eles;
- Desenvolvedores cooperam constantemente e trabalham junto com os usuários/clientes;

Princípios Básicos

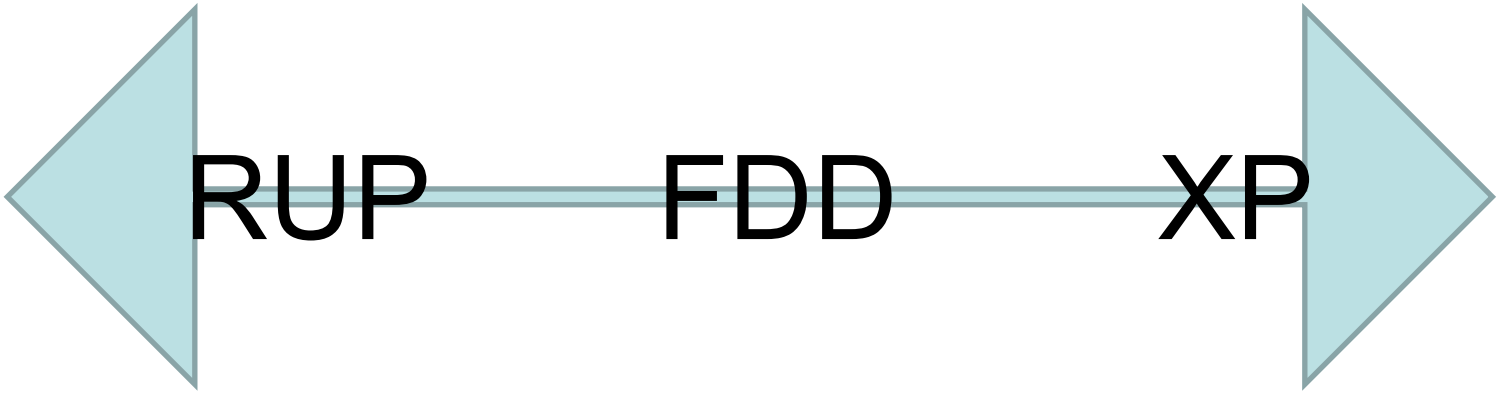
- Atender o usuário/cliente, entregando rapidamente e continuamente produtos funcionais em curto espaço de tempo (normalmente a cada 2 semanas);
- Software funcionando é a principal medida de progresso;
- Mudanças no escopo, ou nos requisitos, do projeto não é motivo de chateação;
- A equipe de desenvolvimento se auto-organiza, fazendo ajustes constantes em melhorias.

Manifesto Ágil

- Esse Manifesto ocorreu para ser um contraponto as Metodologias de Desenvolvimento Prescritivas. Ou seja, enquanto o RUP, é extremamente rígido com altos níveis de controle, e forte documentação, as metodologias ágeis caminham “ao contrário’. Mesmo assim, não inflige a uma sólida prática da Engenharia de Software.

- Assinado por 17 desenvolvedores em Utah em fevereiro/2001.

Metodologias



CONTROLE
Equipes
Grandes
Alto Rigor

ACOMPANHAR
Equipes Médias
e Pequenas

LIBERDADE
Equipes pequenas
(menores de 10)

No gráfico vemos num extremo o RUP enfatizando os controles, e uma política de trabalho rígida. Ele é mais interessante de ser utilizado com equipes grandes de desenvolvimento. Na outra ponta temos o XP sinalizando maior liberdade e mais adequada para equipes pequenas. E num ponto intermediário o FDD é como um modelo conciliador dessas duas estratégias.

Parênteses na disciplina (FDD)

A **FDD** (Feature-Driven Development) é uma metodologia de desenvolvimento de software que, seguindo os princípios propostos pelo Manifesto Ágil, fornece processos para a distribuição repetível de software com valor para o cliente.

Com surgiu a FDD?
Ela surgiu no ano de 1997 quando Peter Coad e Jeff De Luca foram contratados para salvar um projeto bancário em Singapura. Reunindo experiências anteriores, eles chegaram ao que hoje é a FDD, esse mesmo projeto deu ainda origem à técnica de modelagem da UML em cores. Após pouco mais de um ano, o projeto estava salvo, tendo mais de 2.000 features (funcionalidades) desenvolvidas por uma equipe de 50 pessoas.

Identificando a FDD
A FDD possui características marcantes, entre elas podemos citar a importância que é dada para a qualidade das funcionalidades entregues ao cliente, através de práticas como a inspeção de modelo e de código. Outra característica não menos importante é a de priorizar a entrega de resultados frequentes, tangíveis e funcionais para os clientes, através do trabalho dividido em iterações, o que aliás é uma prática muito usada no mundo do desenvolvimento ágil. Relatórios de estado e progresso das atividades (como o famoso Parking Lot), adaptabilidade para projetos e equipes maiores ou menores, e um desenvolvimento partindo de um modelo abrangente são outras fortes características da FDD.

FDD significa Desenvolvimento Dirigido à Funcionalidades, mas...o que é uma funcionalidade?
É alguma característica do sistema que ofereça valor para o cliente e que possa ser desenvolvida em, no máximo, duas semanas. Para os mais experientes, uma funcionalidade pode ser comparada a um requisito funcional.
O template de uma funcionalidade é: [ação] [resultado] [objeto]
Alguns exemplos de funcionalidades: Calcular o desconto de uma venda, Validar o CPF de um cliente, Ordenar por Cidade e UF o relatório de clientes,...

Metodologia Ágil

Um dos pontos de destaque na Metodologia Ágil é a liberdade dada para as equipes de desenvolvimento.

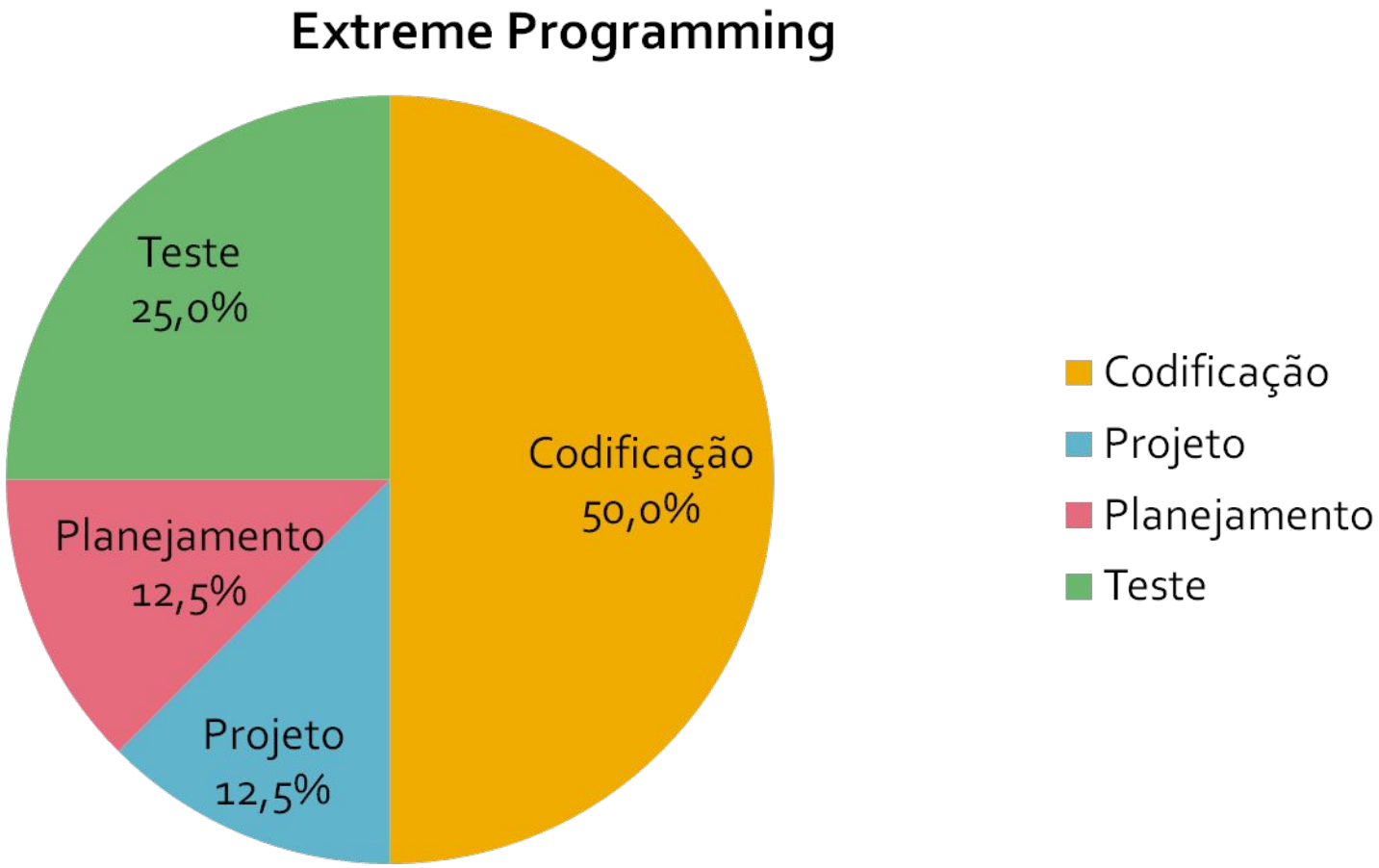
*"A equipe seleciona quanto trabalho acredita que pode realizar dentro da iteração, e a equipe se compromete com o trabalho. Nada desmotiva tanto uma equipe quanto alguém de fora assumir compromissos por ela. Nada motiva tanto uma equipe quanto a aceitação das responsabilidades de cumprir os compromissos que ela própria estabeleceu". (*Ken Schwaber)

XP Extreme Programming

- A Programação Extrema é uma das metodologias ágeis mais conhecidas. Foi criada por Kent Beck.

- Baseada em cinco **valores,** alguns **princípios** e várias **práticas** que ocorrem no contexto de quatro atividades. Ela se destina a times de até dez programadores, projetos de curto e médio prazo.

XP – Fases


Extreme Programming
Teste
25,0%
Codificação
50,0%
Planejamento
12,5%
Projeto
12,5%
Codificação
Projeto
Planejamento
Teste

XP – Cinco Valores

Comunicação – para um projeto de sucesso é necessária muita interação entre os membros da equipe, programadores, cliente, treinador. Para desenvolver um produto, o time precisa ter muita qualidade nos canais de comunicação. Conversas presenciais são sempre melhores do que telefonemas, e-mails, cartas ou fax.

Feedback – as respostas às decisões tomadas devem ser rápidas e visíveis. Todos devem ter, o tempo todo, **consciência** do que está acontecendo.

Coragem – alterar um código em produção, sem causar bugs, com agilidade, exige muita coragem e responsabilidade.

Simplicidade – para atender rapidamente às necessidades do cliente, quase sempre um dos valores mais importantes é simplicidade. Normalmente o que o cliente quer é muito mais simples do que aquilo que os programadores constroem.

Respeito – todos têm sua importância dentro da equipe e devem ser respeitados e valorizados. Isso mantém o trabalho energizado.

XP – Quatro Papeis

Programadores - foco central da metodologia, sem hierarquia.

Treinador (ou coach) - pessoa com mais experiência no time, responsável por lembrar os outros das regras do jogo (que são as práticas e os valores de XP). O treinador não precisa necessariamente ser o melhor programador da equipe e sim o que mais entende da metodologia XP.

Acompanhador (ou tracker) - responsável por trazer para o time dados, gráficos, informações que mostrem o andamento do projeto e ajudem a equipe a tomar decisões de implementação, arquitetura e design. Algumas vezes o próprio coach faz papel de tracker. Outras o time escolhe sozinho quem exercerá este papel.

Cliente – em XP o cliente faz parte da equipe. Deve estar sempre presente e pronto para responder às dúvidas dos programadores.

XP – Práticas

Testes - todo desenvolvimento inclui testes. Kent Beck diz que código sem teste não existe. Os testes devem ser escritos de preferência antes do desenvolvimento (TDD - test driven development) e sempre devem rodar de forma automatizada.

Refatoração - é um conjunto de técnicas para modificar o código do sistema sem alterar nenhuma funcionalidade. O objetivo é simplificar, melhorar o design, limpar, enfim, deixar o código mais fácil de entender e dar manutenção.

Programação Pareada (pair programming) - em XP dois programadores sentam juntos no mesmo computador e programam juntos. Enquanto um programador digita, o outro observa, pensa em melhorias, alternativas.

Propriedade Coletiva - O código fonte não pertence a um único programador. Todos da equipe são responsáveis. Todos alteram código de todos (mas sempre rodando os testes para se certificar que nada foi quebrado)
ção.

Integração Contínua - depois de testada, cada nova funcionalidade deve ser imediatamente sincronizada entre todos os desenvolvedores. Quanto mais freqüente for essa integração, menores são as chances de conflitos de arquivos que vários programadores alteram simultaneamente.

XP – Práticas

Semana de 40 horas - programar é uma atividade intensa e que não rende se o programador não estiver descansado e disposto. Por isso, 40 horas de trabalho por semana é essencial para a saúde do time.

Cliente Sempre Presente - o cliente não é alguém de fora, mas sim um membro da equipe. Ele deve estar sempre disponível e pronto para atender às dúvidas dos desenvolvedores.

Padronizações - se todo o time seguir padrões pré-acordados de codificação, mais fácil será manter e entender o que já está feito. O uso de padrões é uma das formas de reforçar o valor **comunicação**.

Fatores de Qualidade Aplicados ao SCRUM

- **Compatibilidade**: cada incremento deve ser testado com tudo que já foi concluído. Nesse sentido, iterações rápidas e o desenvolvimento em pequenos times favorecem a criação de componentes compatíveis;

- **Corretude**: uso de boas práticas desde o começo. O cliente sempre está presente, verificando se o andamento está de acordo com os requisitos declarados no backlog. Exemplo: verificações realizadas nas reuniões de revisão entre os sprints;

- **Efetividade de Custo**: gráficos burndown auxiliam o acompanhamento dos gastos ao longo do tempo do projeto;

- **Eficiência**: o compartilhamento contínuo de informações entre os membros da equipe favorecem o uso de artefatos que irão otimizar os recursos de hardware e software;

- **Extensibilidade**: na reunião de release, os objetivos são traçados ecaso tenha surgido algum novo item no backlog, as prioridades são alteradas e o escopo do sprint redefinido;

Fatores de Qualidade Aplicados ao SCRUM

- **Facilidade de uso**: apesar da interação constante com o cliente, a metodologia Scrum não se mostra muito adequada para lidar com requisitos de usabilidade.

- **Integridade**: a cada reunião de planejamento cada integrante do time compartilha as lições aprendidas – de sucesso ou não. Seguindo a característica empírica do Scrum, o grupo adota novas práticas ao longo do processo;

- **Manutenibilidade**: à medida que há uma interação maior da equipe em cada Sprint, os erros são detectados rapidamente, assim como é possível projetar software com maior atenção a integração modularidade;

- **Oportunidade**: ao final de cada sprint é realizada a entrega de uma funcionalidade do produto. À medida que ocorrem os sprints, a lista do backlog é reduzida e pode ser modificada a prioridade da lista de acordo com as necessidades do cliente;

- **Portabilidade**: o Scrum adota padrões em todas as fases de desenvolvimento do produto, isso permite a criação de arquiteturas independentes de hardware;

Fatores de Qualidade Aplicados ao SCRUM

- **Reusabilidade**: a integração de componentes é facilitada a partir da larga comunicação no time. As práticas desenvolvidas durante um sprint podem ser utilizadas nas próximas iterações a partir das reuniões diárias, revisão e de retrospectiva;

- **Robustez**: enquanto o produto existir, o backlog também existirá. O dono do produto acompanha a execução de todas as iterações e sempre oferece feedback
sobre o atendimento dos requisitos. Situações não previstas são tratadas rapidamente;

- **Verificação e validação**: reuniões diárias permitem conferir o andamento d trabalho realizado por cada integrante do time. Como cada iteração significa o entregável ao cliente, o produto é verificado, validado e testado constantemente.

Fatores de Qualidade Aplicados aos Métodos Ágeis

• **Facilidade de uso**: casos de uso são desenvolvidos colaborativamente com feedback do cliente ao longo do projeto, proporcionando ajustes para adequar a usabilidade;

• **Integridade**: o gerente de projeto trabalha em conjunto com todaa equipe a fim de identificar a alta prioridade de resolver os itens de trabalho da lista de itens. O plano de iteração anterior incluir uma avaliação dos resultados e também pode ser usado como entrada para o planejamento da iteração atual. No âmbito deste processo, o gerenciamento de configurações mantém as versões de artefatos e configurações consistentes;

• **Manutenibilidade**: gerenciamento de mudança fornece dados para medir o progresso e proporciona um meio eficaz para se adaptar as mudanças e problemas. As versões de todo trabalho são mantidas atualizadas e as alterações são geridas por tarefas de solicitação de mudança que são posteriormente priorizadas em uma lista de itens de trabalho;

Fatores de Qualidade Aplicados aos Métodos Ágeis

• **Oportunidade**: atende este fator por realizar desenvolvimento iterativo e incremental, permitindo entrega rápida e realizando curtos ciclos;

• **Portabilidade**: o uso de padrões de desenvolvimento praticado no mercado separa o software em camadas permitindo a independência e autonomia de hardware e software nesta metodologia;

• **Reusabilidade**: com o apoio de documentação adequada é possível fazer o reuso de código e adaptar de forma simples a diferentes processos e aplicações, reduzindo o tempo de desenvolvimento e garantindo este fator de qualidade;

• **Robustez**: ao longo das iterações a disciplina arquitetura se relaciona com outras disciplinas, como requisitos ao obter arquitetura significativa, testes ao verificar a estabilidade e corretude, entre outras, para alcançar uma arquitetura robusta para o sistema;

• **Verificação e validação**: testadores são responsáveis por realizar testes de funcionalidade, usabilidade, confiança, performance e suportabilidade várias vezes por iteração, cada vez que a solução é incrementada com o desenvolvimento de algo novo, mudança ou correção de erro, no sentido de retirar os riscos mais cedo no ciclo do sistema e garantir a qualidade de software.

Métodos Ágeis e Qualidade de Sofware

A qualidade é uma atividade presente nas diferentes metodologias ágeis de desenvolvimento de software e os princípios relatados no manifesto ágil norteiam também a busca pela qualidade.

A abordagem ágil modificou a forma de desenvolvimento de software. As metodologias ágeis também mudaram a forma de atividades de SQA. Documentações não são muito pesadas, mas apenas o que o cliente/usuário necessita

Muitas características já incorporadas na filosofia ágil tem um potencial de garantir a qualidade do software produzido. Tais como a **Refatoração, Test-Driven Development (TDD), Programação em Par**

Fontes

MNKANDLA, Ernest; DWOLATZKY, Barry. Defining Agile Software Quality Assurance. Proceedings of the International Conference on Software Engineering Advances, p. 36-36, 2006.

BECK, Kent; BEEDLE, Mike; BENNEKUM, Arie van; COCKBURN, Alistair; CUNNINGHAM, Ward; FOWLER, Martin; GRENNING, James; HIGHSMITH, Jim; HUNT, Andrew; JEFFRIES, Ron; KERN, Jon; MARICK, Brian; MARTIN, Robert C.; MELLOR, Steve; SCHWABER, Ken; SUTHERLAND, Jeff; THOMAS, Dave. Agile Manifesto. 2001. Disponível em: < http://agilemanifesto.org>.

SCHWABER, Ken. Agile Project Management with Scrum.Redmond: Microsoft Press, 2004.

SCHWABER, Ken; SUTHERLAND, Jeff. Scrum Guide. 2010.Disponível em < http://www.scrum.org/scrumguides/ >

BIBLIOGRAFIAS

BIBLIOGRAFIA

KOSCIANSKI, A.; SOARES, M. S. “Qualidade de Software”, São Paulo, Editora Novated, 2006.
PRESSMAN, R.S., Engenharia de Software. 6a edição, McGrawHill, 2006.

ROCHA, A.R., Weber, K., MALDONADO, J.C., Qualidade de Software: Teoria e Prática. Prentice Hall, 2001.

Bibliografia Complementar

BARTIÉ, A.”Garantia da qualidade de software”., Rio de Janeiro, Campus, 2002.

MPS.BR - Melhoria de Processo do Software Brasileiro – Guia Geral, Softex, 2009.

SOMMERVILLE, I.”Engenharia de software”. 8. Ed. São Paulo: Addison-Wesley, 2007.

BECK, K. Programação Extrema (xp) Explicada – Acolha as Mudanças. Ed. Bookman, 2004.
Guide to the Software Engineering Body of Knowledge, IEEE Computer Society, 2004. Disponível em http://swebok.org.

SEI, Software Engineering Institute, Carnegie Melon University, http://www.sei.cmu.edu.

Qualidade de Produto de Software - http://www.mct.gov.br/index.php/content/view/306537.html

KRUCHTEN, P. Rational unified process made easy. Boston: Addison-Wesley Professional, 2003, p. 37.