Fatec - Curso: Análise e Desenvolvimento de Sistemas
Disciplina – Teste de Software
Prof. Edson Saraiva de Almeida

Atividades de Controle da Qualidade no Processo de Desenvolvimento de Software

Objetivos de aprendizagem: Reconhecer o fato de que modelos de processo de desenvolvimento de software devem ser adaptados para o contexto do projeto e características do produto. Entender o relacionamento entre atividades de desenvolvimento e controle da qualidade. Aplicar atividades de controle da qualidade considerando o contexto do projeto e as características do produto.

1. Introdução

O primeiro passo, para organizar as atividades que devem ser realizadas em uma empresa na produção de software, é estabelecer o Processo de Desenvolvimento de Software (PDS). Para o sucesso no estabelecimento e na evolução dos processos de software, além de conhecer os métodos e processos de engenharia de software, é fundamental que outros aspectos sejam considerados. Por exemplo, as características do produto, do projeto, a cultura organizacional, as necessidades estabelecidas pelo cliente, o ambiente, as tecnologias utilizadas, devem ser consideradas.

Um PDS estabelece um conjunto de atividades que devem ser seguidas na construção do software. Um requisito básico de qualidade relacionado ao processo de desenvolvimento é que a atividade deve ser sistemática e passível de repetição, independentemente de quem as execute. Quanto ao produto de software, é também fundamental que sua qualidade seja independente de quem a produziu.

2. Modelos de Processo

Modelos de processos são prescritivos, definem um conjunto de atividades, marcos e produtos de trabalho para permitir a entrega de software com qualidade. Podem ser adaptados, para atender as necessidades da equipe em projetos específicos. Existem normas e modelos de qualidade que apoiam a definição de processos de desenvolvimento de software a NBR ISO/IEC 12207 (ISO,1998) é um exemplo. Atividade é um passo do processo que produz mudanças de estado visíveis externamente ao produto de software. Exemplos de atividades seriam teste do software, elicitação de requisitos, implementação, etc. Produto de trabalho é um artefato produzido ou consumido em uma atividade.

Nos modelos de processo preditivos (Cascata) o desenvolvimento do projeto é linear através de fases sequenciais distintas, os objetivos são determinados no início do projeto e só podem ser alterados através de um controle formal. Esse modelo é particularmente adequado quando o produto deve ser entregue completo para ter valor. Nos modelos de processo interativo e incremental, o produto é desenvolvido por meio de ciclos de interações sucessivas, cada interação inclui novas funcionalidades (incrementos) na versão anterior do produto, obtendo vantagem das lições anteriormente aprendidas. Nestes modelos uma visão de alto nível é desenvolvida para o empreendimento em geral, mas o escopo detalhado é elaborado a cada interação. Em modelos adaptativos ou ágeis o desenvolvimento do trabalho é realizado por meio de pequenos ciclos interativos e incrementais, geralmente com duração e recursos fixos, essa abordagem é particularmente apropriada quando existe indefinição do escopo.

O modelo descritivo de um processo inclui as atividades e eventualmente subatividades e os artefatos gerados como produto de trabalho. A descrição do modelo de processo pode ser realizada utilizando linguagem natural (texto) ou em uma linguagem mais formal, por exemplo, utilizando uma notação gráfica.

1.1 Modelo Cascata

O modelo em cascata, algumas vezes chamado de ciclo de vida clássico, Cascata ou Watefall, sugere uma abordagem sistemática e sequencial para desenvolvimento de software que começa com a especificação dos requisitos pelo cliente e progride até a implantação do software acabado. Devido ao encadeamento de uma fase em outra, esse modelo é conhecido como modelo cascata (Figura 1).

O modelo é caracterizado pelos seguintes aspectos:

- O resultado de cada fase consiste em um ou mais artefatos estabelecidos pela organização
- A próxima fase só é iniciada quando a fase anterior estiver concluída e aprovada

O modelo cascata pode ser descrito utilizando à notação proposta pelo método IDEF0 (Marca e McGowan 1993). Os elementos básicos de um diagrama IDEF0, são apresentados na Figura 1. As caixas representam as atividades; as setas entrando no lado esquerdo da caixa representam as entradas (artefatos) que serão transformadas pela atividade para produzir os resultados, representados pelas setas do lado direito da caixa. As setas entrando pela borda superior da caixa são os controles, ou seja, o que dirige ou restringe as atividades. As setas que entram pela borda inferior da caixa representam os mecanismos, que descrevem os recursos (ferramentas) utilizadas pela atividade.

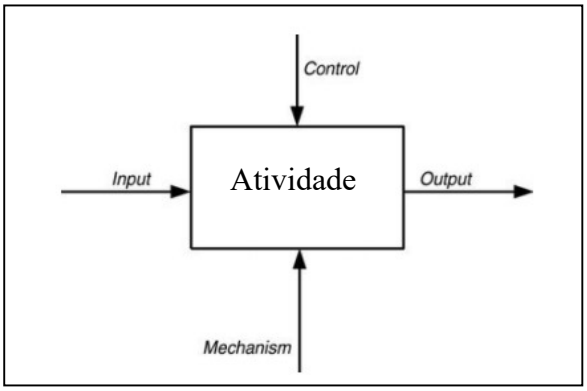

Figura 1 IDEF0 (Marca, 1993)

A Figura 2 apresenta um exemplo de um processo que segue o modelo Cascata (Waterfall) considerando os critérios de aceitação por fase utilizando o IDEF0 para descrever o processo.

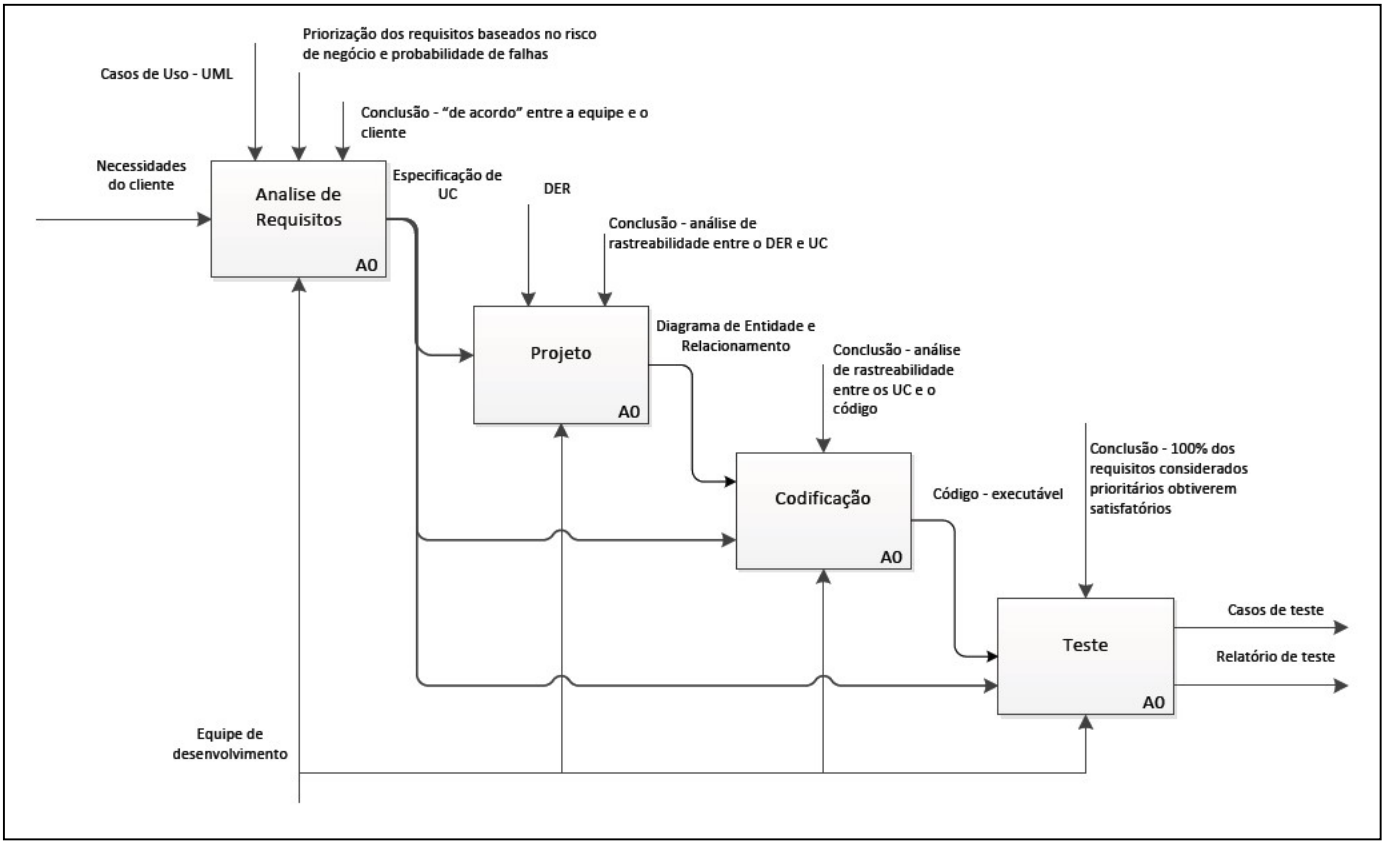

Figura 2 – Exemplo de descrição para um processo de desenvolvimento de software utilizando o IDEF0

Detalhamento do diagrama IDEF0 para o processo da Figura 2:

Atividade - Análise de Requisitos	
Entrada	Necessidades identificadas pelo cliente
Regras – estabelece como a qualidade será atingida	1 – os requisitos serão documentados utilizando o modelo de casos de uso da UML
	2 – os requisitos serão priorizados de acordo com a criticidade para o negócio e a probabilidade de falhas
	3 - esta fase será considerada concluída quando for realizada a reunião entre a equipe de desenvolvimento e o cliente para estabelecer o "de acordo" no documento de requisitos
Saída	Diagrama de casos de uso e especificação de casos de uso
Mecanismos	A atividade é de responsabilidade da equipe de desenvolvimento

Atividade – Projeto	
Entrada	Diagrama de casos de uso e especificação de casos de uso
Regras – estabelece como a qualidade será atingida	1 – nesta atividade o diagrama de entidade e relacionamento utilizando a notação "crows foot" do MySQL Workbench é gerado.
	2 – esta atividade será considerada concluída quando a análise de rastreabilidade entre o DER – Diagrama de Entidade e Relacionamentos e o documento de requisitos não identificarem inconsistências
Saída	Diagrama de Entidade e Relacionamentos
Mecanismos	A atividade é de responsabilidade da equipe de desenvolvimento

Atividade - Codificação	
Entrada	1 - Diagrama de casos de uso e especificação de casos de uso
	2 – Diagrama de entidade e relacionamento
Regras – estabelece como a qualidade será atingida	Será considerada concluída quando a análise de rastreabilidade entre o código e as funções solicitadas no documento de requisitos não identificarem inconsistências
Saída	Código executável
Mecanismos	A atividade é de responsabilidade da equipe de desenvolvimento

Atividade – Teste	
Entrada	1 – Especificação de UC 2 - Código executável
Regras – estabelece como a qualidade será atingida	Será considerada concluída quando 100% dos casos de teste prioritários rastreáveis para os requisitos obtiverem satisfatório
Saída	1 – Casos de Teste 2 - Relatório de Teste
Mecanismos	A atividade é de responsabilidade da equipe de desenvolvimento

Vantagens do modelo cascata:

- Visão de alto nível - apresenta uma visão de alto nível e sugere aos desenvolvedores a sequência de eventos esperada para a entrega do software.
- Desenvolvimento em estágios - o desenvolvimento de um estágio deve terminar antes do próximo começar
- Produtos intermediários - explicita os produtos intermediários necessários para começar o próximo estágio de desenvolvimento.

- Marcos – fornecem marcos bem definidos associados a cada atividade do processo, os gerentes podem se utilizar do modelo para estabelecer os critérios de qualidade que definem a conclusão de cada fase (marco), a decomposição por fases também facilita a estimativa do término do projeto.

Desvantagens do modelo cascata:

- Fluxo sequencial - projetos reais raramente seguem o fluxo sequencial que o modelo propõe
- Mudanças - o modelo tem dificuldades para acomodar as incertezas naturais que existem no começo do projeto
- Risco - uma versão executável do programa não vai ficar disponível até o projeto terminar – um erro grosseiro pode ser desastroso.
- Estados de bloqueio - alguns membros da equipe precisam esperar que outros membros completem tarefas dependentes.

Deve ser utilizado quando os requisitos forem bem compreendidos e houver pouca probabilidade de mudanças radicais durante o desenvolvimento do sistema (PRESSMAN, 2006; SOMMERVILLE, 2007). Este cenário ocorre quando o mercado está bem definido, o domínio onde o software será utilizado está estabelecido os clientes são conhecidos.

1.2 Modelo V

O Modelo V foi desenvolvido para minimizar alguns dos problemas identificados na utilização da modelo cascata tradicional. Os defeitos eram encontrados nas fases finais do ciclo de desenvolvimento. No Modelo V o teste é iniciado o mais cedo possível no ciclo de vida. O modelo estabelece uma série de atividades que devem ser executadas antes da codificação. Estas atividades devem ser executadas em paralelo com o ciclo de desenvolvimento do software. Existem diversos fatores que podem colaborar para ocorrência de erros, por isso, a atividade de testes é dividida em fases com objetivos distintos (Figura 3). De uma forma geral pode-se dividir a atividade de teste nas seguintes fases: teste de unidade, teste de integração e teste de sistemas.

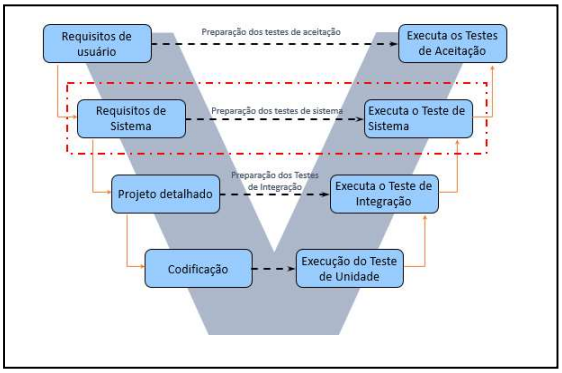

Figura 3- Modelo V

Vantagens – atividades de teste como planejamento e projeto acontecem antes da codificação iniciar, defeitos são encontrados em fases iniciais do ciclo de desenvolvimento de software.

Desvantagens – semelhantes a do modelo cascata, pouco flexível a mudanças nos requisitos

Critérios de seleção – este modelo pode ser selecionado com objetivos de melhoria do processo, quando a organização está enfrentando problemas com o modelo cascata, incluindo atividades de controle da qualidade que são realizadas em todas as fases do processo de desenvolvimento antecipando a descoberta de defeitos, ou quando o cliente exige evidencias da realização das atividades de teste nas fases intermediarias.

1.3 Desenvolvimento de software incremental

Atualmente o trabalho de software é executado em ritmo rápido. Mudanças frequentes sujeitas a um volume sem fim de modificações. O modelo cascata é frequentemente inadequado para este tipo de trabalho. O aumento da demanda e a pressão por prazos exigiu novos modelos de processo para ajudar a reduzir o tempo de desenvolvimento. Em um processo de entrega incremental um esboço dos requisitos é identificado pelos clientes e divididos em vários incrementos. Cada incremento proporciona um subconjunto da funcionalidade do sistema (Figura 4).

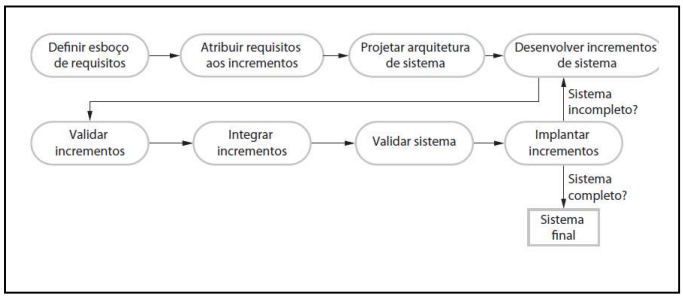

Figura 4- Processo de desenvolvimento iterativo (SOMMERVILLE, 2007)

O cliente identifica, em linhas gerais, os serviços a serem fornecidos pelo sistema, a atribuição de serviços aos incrementos depende da ordem de prioridade dos serviços — os serviços de mais alta prioridade são implementados e entregues em primeiro lugar. Uma vez que os incrementos do sistema tenham sido identificados, os requisitos dos serviços a serem entregues no primeiro incremento são definidos em detalhes, e esse incremento é desenvolvido. Durante o desenvolvimento, podem ocorrer mais análises de requisitos para incrementos posteriores, mas mudanças nos requisitos do incremento atual não são aceitas.

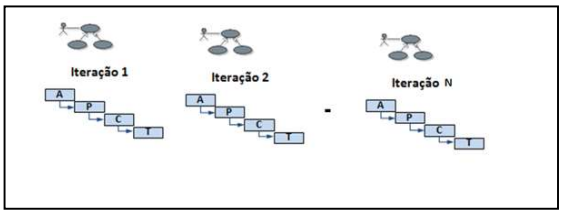

Figura 5 - Cada iteração pode ser considerada um mini-cascata

Cada iteração produz software funcional diferentemente da modelo cascata (tradicional). Existe verificação a cada etapa para obter confirmação do cliente. Para facilitar a compreensão pode-se pensar na iteração com uma analogia do modelo cascata, cada iteração pode ser considerada um mini-cascata (Figura 5).

Critérios de seleção – Os principais requisitos estão definidos mais alguns detalhes podem evoluir com o tempo, esta abordagem permite diminuir os riscos antes que se faça muitos investimentos no projeto. Este modelo de processo é adequado quando existe necessidades de entregas parciais, disponibilização um retorno mais rápido para o cliente, liberando software integrado e testado que pode ser instalado no ambiente de produção.

2. Scrum

Scrum é um processo ágil que permite manter o foco na entrega do maior valor de negócio, no menor tempo possível (Figura 7). Esta abordagem permite entregas rápidas e inspeção contínua do software em produção (em intervalos de duas a quatro semanas). As necessidades do negócio é que determinam as prioridades do desenvolvimento do software. As equipes se auto organizam para definir a melhor maneira de entregar as funcionalidades de maior prioridade. O Scrum é fundamentado nas teorias empíricas de controle de processo.

O coração do Scrum é a Sprint, que tem duração de 2 a 4 semanas durante o qual um “Pronto” (versão incremental potencialmente utilizável do produto) é criado. As Sprints são compostas por uma reunião de planejamento da Sprint, reuniões diárias, o trabalho de desenvolvimento, uma revisão da Sprint e a retrospectiva da Sprint. A reunião de planejamento da Sprint responde as seguintes questões: (1) Qual é o objetivo da Sprint? (2) O que pode ser entregue como resultado do incremento da próxima Sprint? (3) Como o trabalho necessário para entregar o incremento será realizado?

A Reunião Diária é mantida no mesmo horário e local para reduzir a complexidade logística. Durante a reunião os membros do Time de Desenvolvimento respondem três perguntas: (1) O que eu fiz ontem que ajudou o Time de Desenvolvimento a atender a meta da Sprint? (2) O que eu farei hoje para ajudar o Time de Desenvolvimento atender a meta da Sprint? (3) Eu vejo algum obstáculo que impeça a mim ou o Time de Desenvolvimento no atendimento da meta da Sprint?

O Time Scrum é composto pelo Product Owner, o Time de Desenvolvimento e o Scrum Master. Times Scrum são auto organizáveis e multifuncionais. Times auto organizáveis escolhem qual a melhor forma para completarem seu trabalho, em vez de serem dirigidos por outros de fora do Time. Times multifuncionais possuem todas as competências necessárias para completar o trabalho sem depender de outros que não fazem parte da equipe. O modelo de time no Scrum é projetado para aperfeiçoar a flexibilidade, criatividade e produtividade. Os artefatos mantidos pelo Scrum são o backlog de produto e backlog da Sprint.

.

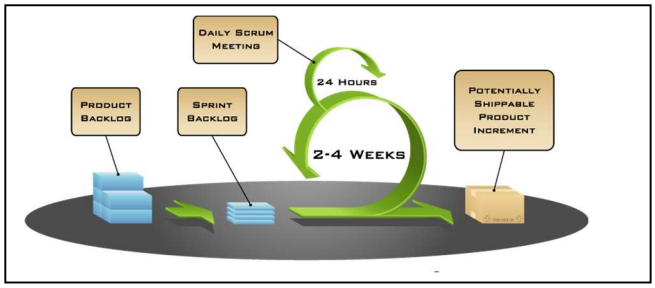

Figura 6 - Cerimônia do Scrum

O Sucesso no uso do Scrum depende das pessoas se tornarem mais proficientes na vivência dos cinco valores Scrum comprometimento, coragem, foco, transparência e respeito. As pessoas se comprometem pessoalmente em alcançar estes objetivos do Time Scrum. O Time Scrum precisa ter coragem para fazer a coisa certa e trabalhar em problemas difíceis. Todos focam no trabalho da Sprint e nos objetivos do Time Scrum. O Time Scrum e seus stakeholders concordam em estarem abertos a todo o trabalho e aos desafios com a execução dos trabalhos. Os membros do Time Scrum respeitam uns aos outros para serem pessoas capazes e independentes.

2.1 Controlando a qualidade

Três pilares apoiam a implementação de controle de processo no Scrum: transparência, inspeção e adaptação. Manter a transparência implica que aspectos significativos do processo devem estar visíveis aos responsáveis pelos resultados. Esta transparência estabelece um padrão comum para que os observadores compartilharem um mesmo entendimento das fases do processo. Por exemplo, uma linguagem comum que estabelece o processo de desenvolvimento, deve ser compartilhada por todos os participantes.

Os usuários Scrum devem frequentemente inspecionar os artefatos Scrum e o progresso com objetivo de detectar desvios. Esta inspeção não deve ser tão frequente que atrapalhe a própria execução das tarefas. As inspeções são mais benéficas quando realizadas de forma cuidadosa por inspetores especializados no trabalho a se verificar. Se um inspetor determina que um ou mais aspectos de um processo desviou para fora dos limites aceitáveis, e que o produto resultado será inaceitável, o processo ou o material sendo produzido deve ser ajustado. O ajuste deve ser realizado o mais breve possível para minimizar mais desvios.

O planejamento da qualidade é realizado a cada iteração durante o desenvolvimento do produto. Na fase de conclusão do Sprint o controle da qualidade é realizado durante a "Revisão do Sprint" e na reunião de Retrospectiva. Na reunião de Retrospectiva o time discute o processo de desenvolvimento para avaliar o que agregou valor e qual aspecto não atendeu as expectativas da equipe. Neste momento, poderão ser feitas sugestões de melhoria que serão revisadas pelo time para o próximo Sprint. Uma definição comum de "Pronto" deve ser compartilhada por aqueles que realizam o trabalho e por aqueles que aceitam o resultado do trabalho. A definição do "Pronto" busca orientar o time para que tenha um entendimento comum a respeito do que significa uma entrega pronta – análise estática do código, testes de unidade, testes de integração e aceitação concluídos.

2.2 Práticas utilizadas em métodos ágeis que podem contribuir para melhorar a qualidade

Uma visão da arquitetura é um elemento chave no desenvolvimento de software mesmo quando construindo a arquitetura sob demanda, nos casos onde o modelo de processo selecionado é iterativo e incremental (PHAM, 2011). Para chegar a uma visão da arquitetura, deve-se primeiro identificar as histórias de usuário que compartilham informações de negócio comuns e considerar construí-las em conjunto (Figura 7).

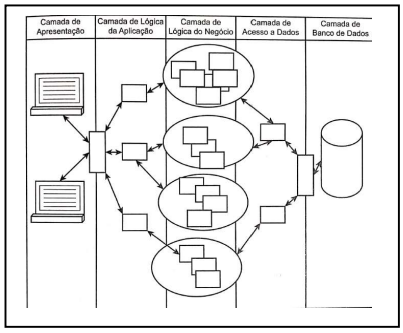

Figura 7 - Visão da arquitetura (PHAM, 2011)

Programação em pares é a prática na qual dois desenvolvedores trabalham juntos na mesma atividade utilizando somente um computador. Enquanto um desenvolvedor codifica, o outro se concentra em atividades mais estratégicas, tais como revisão de código, ou encontrar melhores soluções para atender uma necessidade solicitada pelo cliente. Outra prática que pode contribuir é o desenvolvimento orientado por teste. Dentre outros fatores o TDD ajuda a manter controlado o crescimento do custo das mudanças do projeto, ou seja, mantem a solução mais facilmente modificável durante o desenvolvimento, permitindo revisões constantes e viabilizando a estratégia de adaptação como um todo.

3. Ciclo de avaliação e melhoria de processos

Um outro aspecto fundamental, ao se definir o processo de desenvolvimento de software em uma organização, está relacionado ao estabelecimento de um processo de evolução do processo de desenvolvimento, de modo que essa evolução seja sistemática e disciplinada. Atualmente, existe uma procura constante da sociedade por softwares mais baratos e melhores, consequentemente, muitas empresas de desenvolvimento de software voltaram-se para a melhoria de processos de software como uma forma de melhorar a qualidade dos seus produtos, reduzindo custos ou acelerando seus processos de desenvolvimento. A melhoria de processos implica a compreensão dos processos existentes e sua mudança para aumentar a qualidade de produtos e/ou reduzir custos e o tempo de desenvolvimento. Duas abordagens podem ser utilizadas:

•A abordagem de maturidade de processo, a qual se centra em melhorar o gerenciamento de processos e projetos e em introduzir boas práticas de engenharia de software em uma organização. O nível de maturidade de processos reflete o grau em que as boas práticas técnicas e de gerenciamento foram adotadas nos processos de desenvolvimento de software da organização. Os principais objetivos dessa abordagem são produtos de melhor qualidade e previsibilidade de processo.

•A abordagem ágil, a qual se centra no desenvolvimento iterativo e na redução de overheads gerais no processo de software. As principais características dos métodos ágeis são a entrega rápida de funcionalidade e a capacidade de resposta às mudanças de requisitos de cliente.

Referencias

(SOMMERVILLE, 2011) SOMMERVILLE, I., Engenharia de Software, 9ed., São Paulo: Pearson Prentice Hall, 2011 (Cap. 2, 26)

(ISO, 1998) NBR ISO / IEC 12207:1995. Tecnologia de informação-Processos de ciclo de vida de software - Rio de Janeiro: ABNT, 1998

Marca, D., and McGowan, C., IDEF0 - SADT Business Process and Enterprise Modelling, Ecletic Solutions Corporation, 1993.

PHAM, Andrew; PHAM, Phuong-Van. SCRUM em ação. Sao Paulo: Novatec Editora, 2011.