METODOLOGIA DE DESENVOLVIMENTO DE SISTEMAS

Uma metodologia de desenvolvimento de software é um conjunto de atividades que auxiliam a produção de software. O resultado dessas atividades é um produto que reflete a forma como todo o processo foi conduzido. Pode ser caracterizada como um *framework* de processos organizacionais que, se usados adequadamente, garantem o sucesso da área de Engenharia de Software.

Conforme CARVALHO (2001), "uma metodologia de desenvolvimento detalha as atividades do ciclo de vida, especificando um conjunto único e coerente de princípios, métodos, linguagem de representação, normas, procedimentos e documentação, que permitem ao desenvolvedor de software implementar sem ambigüidade as especificações advindas das fases do ciclo de vida do software".

Embora tenham sido criadas várias metodologias para o desenvolvimento de software, existem atividades fundamentais comuns a todas elas [SOMMERVILLE, 2003]:

- Especificação: definição das funcionalidades e demais características do produto.
- Projeto e implementação: o software é produzido de acordo com as especificações. Nesta fase são propostos modelos por meio de diagramas que serão implementados em alguma linguagem de programação.
- Validação: atividades de revisão e testes visando a assegurar que os requisitos sejam cumpridos.
- Evolução: atividades de manutenção, por exemplo, para adaptar o software a novas necessidades do cliente.

Um ambiente de desenvolvimento de software de qualidade se inicia com uma sólida definição do processo que inclui atividades usualmente definidas como fases, tarefas, passos, e o que será produzido por cada uma dessas atividades. O processo também especifica a ordenação das atividades, que podem ser seqüenciais, concorrentes ou em paralelo, e todas reunidas definem a base da execução do desenvolvimento.

Muitas organizações erradamente confundem o processo com a utilização de certas ferramentas de desenvolvimento [COSTA, 1999, p. 28-30].

Em particular, as pequenas e médias organizações não possuem recursos suficientes para adotar o uso de metodologias pesadas e, por essa razão, normalmente não utilizam nenhum processo. O resultado dessa falta de sistematização na produção de software é a baixa qualidade do produto final, além de dificultar a entrega do software nos prazos e custos predefinidos e inviabilizar a futura evolução do software.

METODOLOGIAS EXISTENTES

O número de metodologias propostas para o desenvolvimento de software atingiu um número demasiado elevado, o que torna virtualmente impossível a sua apresentação. Por

isso, enumeramos algumas metodologias, estruturadas e orientadas a objeto, conhecidas que maior relevância e divulgação tiveram. Para além destas, existiram outras contribuições importantes que não estão incluídas aqui por não apresentarem uma perspectiva integrada de todo o processo de desenvolvimento, mas apenas sugerirem anotações ou técnicas de modelagem.

1 RUP

A Rational Software é uma empresa especializada no oferecimento de soluções para desenvolvimento e implantação de software, conforme informações constantes no site da empresa. Foi constituída por três das maiores autoridades em orientação a objetos que são Grady Booch, James Rumbaugh e Ivar Jacobson.

O RUP (Rational Unified Process) é uma metodologia para gerenciar projetos de desenvolvimento de software que usa o UML como ferramenta para especificação de sistemas. O RUP é composto por um conjunto de disciplinas que fornecem diretrizes para definição das tarefas e para atribuição das responsabilidades. Seu objetivo é garantir a criação de softwares de alta qualidade, que atenda às necessidades expressas pelo cliente e pelos usuários, e às restrições de prazo e custo. O RUP segue as melhores práticas de desenvolvimento de software: desenvolvimento iterativo, gerenciamento de requisitos, arquitetura baseada em componentes, modelagem visual do software, verificação constante da qualidade e controle de mudanças.

As causas dos fracassos da maioria dos projetos de desenvolvimento de software são similares e requerem boas práticas para que sejam evitadas. As causas mais comuns são: gerenciamento informal dos requisitos, não entendimento das necessidades dos usuários, incapacidade de lidar com as mudanças de requisitos, complexidade crescente e excessiva, qualidade ruim, testes insuficientes e baixa performance. O RUP busca resolver todos estes problemas, além de outros. Para isto ele provê as seguintes ferramentas e recursos:

Desenvolvimento iterativo

O objetivo é conduzir o projeto em ondas, ou seja, em iterações. Cada iteração é tratada de forma tradicional, alguns requisitos e riscos mais críticos são abordados, há um pouco de análise, implementação, testes e implantação. Depois há outra iteração, onde novos requisitos são trabalhados, outros riscos são mitigados, há mais análise, implementação, testes e implantação, até que o produto seja concluído.

O intuito da abordagem iterativa é permitir um melhor gerenciamento dos requisitos, facilitando o tratamento das descobertas constantes que ocorrem durante o projeto: há maior facilidade para incluir novas idéias e requisitos ao projeto, o sistema é desenvolvido em incrementos produzindo-se novos artefatos a cada iteração, os riscos mais críticos são mitigados antes e a equipe converge para o objetivo final do projeto, usando processos previsíveis e repetitivos. Com essa abordagem, há dois grandes benefícios:

• permitir que a equipe progressivamente identifique os componentes que vão compor o sistema e decida quais serão desenvolvidos, reutilizados e quais serão comprados;
• a integração não é um “big bang” no final do projeto.

Como muitos riscos do projeto geralmente estão associados às integrações entre os componentes e os subsistemas, este mecanismo permite aumentar significativamente as chances de sucesso.

Gerenciamento de requisitos

O desafio do gerenciamento de requisitos está no fato de que os requisitos são dinâmicos e mudam durante a vida do projeto.

Arquitetura baseada em componentes

A abordagem de desenvolvimento baseado em componentes viabiliza o reuso e a personalização de componentes em larga escala, sejam eles desenvolvidos em casa ou por terceiros. As aplicações são construídas combinando-se várias partes, algumas pertinentes ao domínio do problema e outras para construção do GUI, acesso a dados, comunicação e outras funcionalidades.

Modelagem de software

Um modelo é uma simplificação da realidade que descreve completamente o sistema a partir de certo ponto de vista. O RUP trabalha com os modelos providos pelo UML: Análise, Banco de dados, Caso de Uso, Implantação, Implementação, Negócio, Design e Teste.

Verificação constante da qualidade

A cada iteração o software é testado, num processo de avaliação contínua da qualidade, verficando-se quais cenários falharam e onde, e quais não foram exercitados. A verificação da qualidade requer a criação de testes para cada cenário, que, por sua vez, representam o comportamento esperado do sistema.

Controle de mudanças

O processo de controle de mudanças tem o objetivo de controlar as versões dos artefatos criados e modificados durante o projeto. O desafio é maior quando a equipe é grande, dispersa em muitos locais, com várias plataformas, muitos sistemas e componentes.

Estrutura estática

Os processos definem “quem” está executando “o que” e “quando”. O “quem” está associado aos papéis que as pessoas vão executar no projeto, sendo que uma mesma pessoa pode executar mais de um papel. O papel define as responsabilidades e as tarefas que serão executadas como, por exemplo, analista de sistemas, designer, projetista de testes, dentre outros, sendo que cada papel requer certo conjunto de conhecimentos.

O “o que” está associado às atividades desempenhadas e aos artefatos trabalhados pelas pessoas executando determinados papéis. Cada atividade representa um pacote de trabalho ou tarefa resumo e produz um resultado relevante para o projeto, geralmente associado à criação ou atualização de um artefato.

O “quando” está associado ao sequenciamento das atividades, que devem ser agrupadas e seqüenciadas de modo a produzir resultados de valor para o projeto. Os processos devem ser adaptados às várias fases do projeto, em função dos objetivos das fases.

Estrutura dinâmica

A abordagem iterativa foi concebida para acomodar mudanças de objetivos e estratégias, pertinentes aos projetos de desenvolvimento de software. Algumas das causas: usuário muda de idéia, o problema muda, mudanças técnicas e mudanças de mercado.

FASES

No RUP, o projeto é composto por quatro fases: concepção, elaboração, construção e transição. Ao contrário da abordagem em cascata, as fases não seguem uma seqüência tradicional de requisitos, análise, programação, integração e testes. Estas fases existem em todos os projetos e cada uma termina num marco relevante para o projeto, quando uma decisão deverá ser tomada – continuar o projeto e aprovar os recursos para a próxima fase ou cancelar. Cada fase do projeto tem um conjunto específico de objetivos:

• Concepção: nesta fase, o foco é chegar a um acordo com os stakeholders quanto à visão do sistema e aos objetivos e estimativas das demais fases do projeto

• Elaboração: esta fase é um processo de engenharia, onde o foco está em especificar uma arquitetura robusta e confiável para o sistema e fazer o planejamento para o restante do projeto

• Construção: a fase de construção basicamente consiste num processo de manufatura, onde o foco está na construção do sistema e no gerenciamento de recursos e otimização de tempo, custos e qualidade.

• Transição: o objetivo desta fase é transferir o produto para a comunidade de usuários

PROCESSOS

Os processos são procedimentos compostos de atividades logicamente seqüenciadas e têm objetivos específicos em relação ao projeto. Cada atividade do processo tem a finalidade de criar ou atualizar um ou mais artefatos.

Processos de engenharia

• Modelagem Corporativa: tem o objetivo de entender a estrutura e a dinâmica da organização na qual o sistema será entregue; identificar problemas correntes na organização e possíveis aperfeiçoamentos; assegurar que o cliente, o usuário final e desenvolvedores possuam a mesma compreensão da empresa e produzir os requisitos de sistemas necessários para suportar os objetivos da organização.

• Requisitos: estabelecer e manter o consentimento entre clientes e stakeholders sobre o que o sistema deve fazer; fornecer uma melhor compreensão dos requisitos aos desenvolvedores de sistemas; definir os limites do sistema; fornecer as bases para o planejamento das iterações, estimativa de custo e tempo de desenvolvimento e definir as interfaces do sistema baseado nas necessidades e objetivos dos usuários.

• Análise e Design: transformar os requisitos dentro de um design do que será o sistema; desenvolver uma arquitetura robusta para o sistema e adaptar o design para combinar com o ambiente de implementação, projetar para performance.

• Implementação: preparar a organização do código em termos de implementação de subsistemas, organizados em layers; implementar classes e objetos em termos de componentes (código fonte, binários, executáveis, etc.); testar os componentes desenvolvidos como unidades e integrar os resultados obtidos por implementadores individuais (ou equipes) em um sistema executável.

• Teste: verificar a interação entre os objetos; verificar a integração de todos os componentes de software; verificar se todos os requisitos foram implementados corretamente e verificar os defeitos e assegurar que eles foram tratados antes da entrega do software.

• Implantação: descrever as atividades associadas à verificação do software. O objetivo é tornar o software disponível ao usuário final.

Processos de suporte

• Gerenciamento de Mudança e Configuração: identificar itens de configuração; restringir alterações para aqueles itens; auditar as alterações feitas neles e definir e gerenciar as alterações daqueles itens.

• Gerenciamento de Projeto: fornecer uma estrutura para gerenciamento de projeto de software; fornecer um guia prático para planejamento, recrutamento, execução e monitoramento de projeto e fornecer uma estrutura para o gerenciamento de risco.

• Ambiente: focar as atividades necessárias para configurar o processo para o projeto; descrever as atividades requeridas para desenvolver as guias mestres no
suporte do projeto e fornecer, para a organização de desenvolvimento de software, o ambiente de processos e ferramentas que suportarão a equipe de desenvolvimento.

Fonte: “UMA METODOLOGIA DE DESENVOLVIMENTO DE SISTEMAS PARA UMA EMPRESA DE PLANO ODONTOLÓGICO” - UNIVERSIDADE FEDERAL DA BAHIA - DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO