Engenharia de Software

Engenharia de software é uma área da engenharia e da computação voltada à especificação, desenvolvimento, manutenção e criação de software, com a aplicação de tecnologias e práticas de gerência de projetos e outras disciplinas, visando organização, produtividade e qualidade. Atualmente, essas tecnologias e práticas englobam linguagens de programação, banco de dados, ferramentas, plataformas, bibliotecas, padrões de projeto de software, processo de software e qualidade de software.

Além disso, a engenharia de software deve oferecer mecanismos para se planejar e gerenciar o processo de desenvolvimento de um sistema computacional de qualidade e que atenda às necessidades de um requisitante de software.

Os fundamentos científicos para a engenharia de software envolvem o uso de modelos abstratos e precisos que permitem ao engenheiro especificar, projetar, implementar e manter sistemas de software, avaliando e garantindo suas qualidades. A área que estuda e avalia os processos de engenharia de software, propondo a evolução dos processos, ferramentas e métodos de suporte a engenharia de software é a Engenharia de Software Experimental.

O termo foi criado na década de 1960 e utilizado oficialmente em 1968 na NATO Science Committee. Sua criação surgiu numa tentativa de contornar a crise do software e dar um tratamento de engenharia (mais sistemático, controlado e de qualidade mensurável) ao desenvolvimento de sistemas de software complexos.

Um sistema de software complexo se caracteriza por um conjunto de componentes abstratos de software (estruturas de dados e algoritmos) encapsulados na forma de algoritmos, funções, módulos, objetos ou agentes interconectados, compondo a arquitetura do software, que deverão ser executados em sistemas computacionais.

Os princípios da Engenharia de Software constituem a base dos métodos, tecnologias, metodologias e ferramentas adotadas na prática e que norteiam a prática de desenvolvimento de soluções de software.

Os princípios se aplicam ao processo e ao produto de software se tornando em prática de desenvolvimento de software através da adoção de métodos e técnicas. Geralmente, métodos e técnicas constituem uma metodologia, as quais, são apoiadas pela utilização de ferramentas

Os princípios-chave são:

Rigor e Formalidade;

Separação de Interesses;

Modularidade;

Alta Coesão;

Baixo Acoplamento.

Abstração;

Antecipação a Mudanças;

Generalidade;

Incrementação;

Requisitos de Software.

Considerando o Rigor e Formalidade, deve-se considerar que a engenharia de software é uma atividade criativa, mas que deve ser realizada de maneira sistemática; o Rigor é um complemento necessário a criatividade que visa aumentar a confiança dos desenvolvimentos de software. A Formalidade é o Rigor no seu nível mais elevado.

Exemplos: Análise matemática (formal) da corretude do Programa, Análises sistemáticas de dados de testes, Documentação rigorosa dos passos de desenvolvimento e os passos de gerenciamento bem como a avaliação dos prazos de entrega.

A Separação de Interesses envolve dominar a complexidade, separando os problemas principais e concentrando-se em um de cada vez (dividir e conquistar) suporte a paralelização de atividades e separação das responsabilidades.

Exemplo: Desenvolvimento por fases de maneira incremental (como no processo Ágil) fazendo a separação dos interesses por atividades e respeitando o tempo. Outro exemplo relacionado a um software relaciona-se a manter os requisitos de funcionalidade, performance e interface e usabilidade de usuário em separado.

A Modularidade considera que um sistema complexo pode ser divido em peças mais simples, chamadas de módulos. Um sistema composto pode módulos é chamado de modular. Se faz fundamental que o suporte a separação de interesses seja suportado, quando lidamos com um modulo em específico deve ser possível ignorar os detalhes dos outros módulos da solução.

Cada modulo deve ter alto nível de coesão, sendo entendido como uma unidade significativa, os componentes de um modulo são fortemente relacionados entre si. O baixo acoplamento remete a baixa interação de um modulo com outros do sistema possibilitando que eles sejam compreendidos como unidades em separado.

A Abstração é um conceito que visa a identificação de aspectos importantes de um fenômeno, ignorando os seus detalhes. O tipo de abstração a ser aplicado depende do propósito. Por exemplo: Os botões de um relógio são a sua interface com o usuário, eles podem ser usados como uma abstração para o propósito interno de ajustar o horário, equações que descrevem um circuito (por exemplo, um amplificador) permitem a um designer pensar sobre amplificação de sinal.

Uma abstração deve tornar possível pensar sobre um sistema através do raciocínio sobre os modelos. A abstração pode ser útil para realizar uma estimativa de custos de um projeto de software através de análise de similaridade com projetos passados.

A Antecipação a Mudanças é diretamente relacionada ao suporte a evolução de um software considerando na arquitetura do software aspectos relacionados ao processo de evolução e compatibilidade com mudanças futuras relacionadas ao domínio de aplicação do software.

A Generalidade é um princípio que visa durante a resolução de um problema, descobrir se ele é uma instância de um problema mais geral, no qual a solução pode ser reutilizada em outros casos. O desafio da generalidade está no balanço entre custo e performance.

A Incrementação é relacionada a evolução de um software através de incrementos estruturados. Pode ser realizado através da entrega de subconjuntos de um sistema desde cedo, visando coletar o feedback dos usuários e adicionar funcionalidades de forma incremental. O processo incremental deve focar inicialmente na funcionalidade, para então, pensarmos na performance da solução, naturalmente o protótipo amadurecerá e se tornará um produto.

Áreas de conhecimento

Segundo o SWEBOK (Corpo de Conhecimento da Engenharia de Software), versão 2004, as áreas de conhecimento da Engenharia de Software são:

Requisitos de software

Projeto de software

Construção de software

Teste de software

Manutenção de software

Gerência de configuração de software

Gerência de engenharia de software

Processos de Engenharia de Software

Ferramentas e Métodos de Engenharia de Software

Qualidade de software

Processo de software

Processo de software, ou processo de engenharia de software, é uma sequência coerente de práticas que objetiva o desenvolvimento ou evolução de sistemas de software. Estas práticas englobam as atividades de especificação, projeto, implementação, testes e caracterizam-se pela interação de ferramentas, pessoas e métodos.

SEE e PSEE são os ambientes voltados ao desenvolvimento e manutenção de processos. O projeto ExPSEE é uma continuação dos estudos de processos, principalmente do ambiente PSEE.

Devido ao uso da palavra projeto em muitos contextos, por questões de clareza, há vezes em que se prefira usar o original em inglês design.

Modelos de processos de software

Um modelo de processo de desenvolvimento de software, ou simplesmente modelo de processo, pode ser visto como uma representação, ou abstração dos objetos e atividades envolvidas no processo de software. Além disso, oferece uma forma mais abrangente e fácil de representar o gerenciamento de processos de software e consequentemente o progresso do projeto.

Exemplos de alguns modelos de processo de software;

Modelos ciclo de vida

Sequencial ou Cascata (do inglês waterfall) - com fases distintas de especificação, projeto e desenvolvimento.

Desenvolvimento iterativo e incremental - desenvolvimento é iniciado com um subconjunto simples de Requisitos de Software e iterativamente alcança evoluções subsequentes das versões até o sistema todo estar implementado

Evolucional ou Prototipação - especificação, projeto e desenvolvimento de protótipos.

V-Model - Parecido com o modelo cascata, mas com uma organização melhor, que permite que se compare com outros modelos mais modernos. Principal ponto é que para cada etapa de um lado tem uma verificação do outro, criando um gráfico no formato da letra V com 2 cascatas.

Espiral - evolução através de vários ciclos completos de especificação, projeto e desenvolvimento.

Componentizado - reúso através de montagem de componentes já existentes.

Formal - implementação a partir de modelo matemático formal.

Ágil

RAD

Quarta geração.

Modelos de maturidade

Os modelos de maturidade são um metamodelo de processo. Eles surgiram para avaliar a qualidade dos processos de software aplicados em uma organização (empresa ou instituição). O mais conhecido é o Capability Maturity Model Integration (CMMi), do Software Engineering Institute - SEI.

O CMMI pode ser organizado através de duas formas: Contínua e estagiada.

Pelo modelo estagiado, mais tradicional e mantendo compatibilidade com o CMM, uma organização pode ter sua maturidade medida em 5 níveis:

Nível 1 - Inicial (Ad hoc): Ambiente instável. O sucesso depende da competência de funcionários e não no uso de processos estruturados;

Nível 2 - Gerenciado: Capacidade de repetir sucessos anteriores pelo acompanhamento de custos, cronogramas e funcionalidades;

Nível 3 - Definido: O processo de desenvolvimento de software é bem definido, documentado e padronizado a nível organizacional;

Nível 4 - Gerenciado quantitativamente: Realiza uma gerência quantitativa do processo de software e do produto por meio de métricas adequadas;

Nível 5 - Em otimização: Usa a informação quantitativa para melhorar continuamente e gerenciar o processo de desenvolvimento. Até março/2012, no Brasil, há somente 13 empresas neste nível.

O (MPS.BR), ou Melhoria de Processos do Software Brasileiro, é simultaneamente um movimento para a melhoria e um modelo de qualidade de processo voltada para a realidade do mercado de pequenas e médias empresas de desenvolvimento de software no Brasil. O MPS.BR contempla 7 níveis de maturidade, de A a G, sendo a primeira o mais maduro. Até agosto/2012, no Brasil, há somente 2 empresas neste nível.

Metodologias e métodos

O termo metodologia é bastante controverso nas ciências em geral e na Engenharia de Software em particular. Muitos autores parecem tratar metodologia e método como sinônimos, porém seria mais adequado dizer que uma metodologia envolve princípios filosóficos que guiam uma gama de métodos que utilizam ferramentas e práticas diferenciadas para realizar algo.

Assim teríamos, por exemplo, a Metodologia Estruturada, na qual existem vários métodos, como Análise Estruturada e Projeto Estruturado (muitas vezes denominados SA/SD, e Análise Essencial).

Dessa forma, tanto a Análise Estruturada quanto a Análise Essencial utilizam a ferramenta Diagrama de Fluxos de Dados para modelar o funcionamento do sistema.

Segue abaixo as principais Metodologias e Métodos correspondentes no desenvolvimento de software:

Metodologia Estruturada

Análise Estruturada

Projeto Estruturado

Programação Estruturada

Análise Essencial

SADT

DFD - Diagrama de Fluxo de Dados

MER - Modelo de Entidades e Relacionamentos

Metodologia Orientada a Objetos

Orientação a Objetos

Rational Unified Process (RUP)

Desenvolvimento ágil de software

Feature Driven Development (FDD)

Enterprise Unified Process (EUP)

Scrum (Scrum)

Crystal (Crystal Clear, Crystal Orange, Crystal Orange Web)

Programação extrema (XP)

Outras Metodologias

Microsoft Solution Framework (MSF)

Modelagem

A abstração do sistema de software através de modelos que o descrevem é um poderoso instrumento para o entendimento e comunicação do produto final que será desenvolvido.

A maior dificuldade nesta atividade está no equilíbrio (tradeoff) entre simplicidade (favorecendo a comunicação) e a complexidade (favorecendo a precisão) do modelo.

Para a modelagem podemos citar 3 métodos:

Análise estruturada, criada por Gane & Searson;

Análise Essencial, criada por Palmer & McMenamin e Ed. Yourdon;

UML, criada por Grady Booch, Ivar Jacobson & Jaimes Rumbaugh. É hoje o método mais comum para o paradigma orientado a objetos.

Ferramentas, tecnologias e práticas

A engenharia de software aborda uma série de práticas e tecnologias, principalmente estudadas pela ciência da computação, enfocando seu impacto na produtividade e qualidade de software.

Destacam-se o estudo de linguagem de programação, banco de dados e paradigmas de programação, como:

Programação estruturada

Programação funcional

Programação orientada a objetos

Componentes de Software

Programação orientada a aspecto

Ferramenta

Outro ponto importante é o uso de ferramentas CASE (do inglês Computer-Aided Software Engineering). Essa classificação abrange toda ferramenta baseada em computadores que auxiliam atividades de engenharia de software, desde a análise de requisitos e modelagem até programação e testes.

Os ambientes de desenvolvimento integrado (IDEs) têm maior destaque e suportam, entre outras coisas:

Editor

Debug

Geração de código

Modelagem

Deploy

Testes não automatizados

Testes automatizados

Refatoração (Refactoring)

Gestão de Riscos nos projectos de Software

Uso da Prototipagem na Eng. de Requisitos

Gerência de projetos

A gerência de projetos se preocupa em entregar o sistema de software no prazo e de acordo com os requisitos estabelecidos, levando em conta sempre as limitações de orçamento e tempo.

A gerência de projetos de software se caracteriza por tratar sobre um produto intangível, muito flexível e com processo de desenvolvimento com baixa padronização.

Planejamento

O planejamento de um projeto de desenvolvimento de software inclui:

Análise Econômica de Sistemas de Informações;

organização do projeto (incluindo equipes e responsabilidades);

estruturação das tarefas (do inglês WBS - work breakdown structure);

cronograma do projeto (do inglês project schedule);

análise e gestão de risco;

estimativa de custos.

Essas atividades sofrem com dificuldades típicas de desenvolvimento de software. A produtividade não é linear em relação ao tamanho da equipe e o aumento de produtividade não é imediato devido aos custos de aprendizado de novos membros. A diminuição de qualidade para acelerar o desenvolvimento constantemente prejudica futuramente a produtividade.

A estimativa de dificuldades e custos de desenvolvimentos são muito difíceis, além do surgimento de problemas técnicos. Esses fatores requerem uma análise de riscos cuidadosa.

Além da própria identificação dos riscos, há que ter em conta a sua gestão. Seja evitando, seja resolvendo, os riscos necessitam ser identificados (estimando o seu impacto) e devem ser criados planos para resolução de problemas.

Análise de requisitos

As atividades de análise concentram-se na identificação, especificação e descrição dos requisitos do sistema de software. De acordo com a ISO/IEC/IEEE 24765 requisito é:

(1) condição ou capacidade necessária por um usuário para resolver um problema ou alcançar um objetivo;

(2) condição ou capacidade que deve ser atingida ou possuída por um sistema ou componente de um sistema para satisfazer um contrato, padrão, especificação ou outro documento formalmente imposto;

(3) representação documentada de uma condição ou capacidade como em (1) ou (2);

(4) condição ou capacidade que deve ser alcançada ou possuída por um sistema, produto, serviço, resultado ou componente para satisfazer um contrato, padrão, especificação ou outro documento formalmente imposto. Requisitos incluem as necessidades quantificadas e documentadas, desejos e expectativas do patrocinador, clientes e outras partes interessadas.

Há várias classificações sobre requisitos, o PMBOK e o BABOK utilizam a seguinte classificação hierárquica:

Requisitos de negócio

Requisitos das partes interessadas

Requisitos da solução

Requisitos da transição

Tanto requisitos da solução como da transição se subdividem em: Requisitos funcionais e não funcionais

É comum que o cliente não saiba o que ele realmente deseja, que haja problemas na comunicação e ainda que haja mudança constante de requisitos. Todos esses fatores são recrudescidos pela intangibilidade sobre características de sistemas de software, principalmente sobre o custo de cada requisito.

Estudo de Viabilidade (Levantamento de Requisitos)

A Engenharia de requisitos é um processo que envolve todas as atividades exigidas para criar e manter o documento de requisitos de sistema (SOMMERVILLE). Segundo RUMBAUGH, alguns analistas consideram a Engenharia de requisitos como um processo de aplicação de uma metodologia estruturada combinada com a metodologia orientada a objetos. No entanto, a Engenharia de requisitos possui muito mais aspectos do que os que estão abordados por esses métodos.

Abaixo um pequeno Processo de Engenharia de Requisitos (SOMMERVILLE):

Estudo da viabilidade - "Relatório de Viabilidade"

Obtenção e Análise de Requisitos - "Modelos de Sistema"

Especificação de Requisitos - "Requisitos de Usuário e de Sistema"

Validação de Requisitos - "Documento de Requisitos"

O primeiro processo a ser realizado num Sistema novo é o Estudo de Viabilidade. Os resultados deste processo devem ser um relatório com as recomendações da viabilidade técnica ou não da continuidade no desenvolvimento do Sistema proposto. Basicamente um estudo de viabilidade, embora seja normalmente rápido, deverá abordar fundamentalmente as seguintes questões:

O Sistema proposto contribui para os objetivos gerais da organização?

O Sistema poderá ser implementado com as tecnologias dominadas pela equipe dentro das restrições de custo e de prazo? Ou precisa de treinamentos adicionais?

O Sistema pode ser integrado, e é compatível com os outros sistemas já em operação?

Gestão

Existem cinco tipos de gestão: pessoal, produto, processo, projeto e material.