Processo de desenvolvimento de software

Origem: Wikipédia, a enciclopédia livre.

Um *processo de desenvolvimento de software* é um conjunto de atividades, parcialmente ordenadas, com a finalidade de obter um produto de software. É estudado dentro da área de Engenharia de Software, sendo considerado um dos principais mecanismos para se obter software de qualidade e cumprir corretamente os contratos de desenvolvimento, sendo uma das respostas técnicas adequadas para resolver a Crise do software.

Um outro conceito que podemos utilizar para compreender o processo de desenvolvimento de software foi o apresentado por Waslwick(2013, p 30): é formado por um conjunto de passos de processo parcialmente ordenados, relacionados a artefatos, pessoas, estruturas organizacionais e restrições, tendo como objetivo produzir e manter os produtos de software finais requeridos. Este é um conceito que considera o contexto ao qual o processo de desenvolvimento de software é aplicado destacando as suas interdependências com outros fatores como restrições, pessoas, recursos, padrões etc que impactam no resultado final do processo.

Índice

Histórico

Anos 70

- Programação estruturada desde 1986
- Metodologia de Desenvolvimento de Sistemas (*Cap Gemini SDM* ou *SDM2*)

Anos 80

- Estrutura de Análise de Sistemas e Metodologia de Projeto - *Structured Systems Analysis and Design Methodology (SSADM)*
- Sistema de investigação ou de aprendizagem - *Soft systems methodology (SSM)*

Anos 90

- Programação Orientada a Objetos - *Object-oriented programming (OOP)*
- Desenvolvimento Rápido de Aplicação - *Rapid Application Development (RAD)*, desde 1991
- Metodologia de Desenvolvimento de Sistemas Dinâmicos - *Dynamic Systems Development Method (DSDM)*
- Scrum, desde 1995
- Team Software Process (TSP), desde 1998
- Processo Unificado Rational - *Rational Unified Process (RUP)*
- Programação extrema - *Extreme programming*, desde 1999

Anos 2000

- Processo Ágil Unificado - *Agile Unified Process (AUP)*

Abordagens

- Modelo em cascata (*Waterfall development*)
- Prototipação (*Prototyping*)
- Desenvolvimento incremental (*Incremental development*)
- Desenvolvimento iterativo e incremental (*Iterative and incremental development*)
- Modelo em espiral (*Spiral development*)
- Desenvolvimento Rápido de Aplicação (*Rapid application development - RAD*)
- Desenvolvimento ágil de software (*Agile development*)
- Programar e Arrumar (*Code and fix*)
- Metodologias leves (*Lightweight methodologies*)
- Outras

Passos/Atividades Processo

Análise Econômica

Visa a estabelecer se o projeto de Software gerará lucro, e se a receita gerada será o suficiente para cobrir os custos.

Análise de requisitos de software

A extração dos requisitos de um cliente

Especificação

A especificação é a tarefa de descrever precisamente o software, preferencialmente de uma forma matematicamente rigorosa. Na prática, somente especificações mais bem sucedidas foram escritas para aplicações bem compreendidas e afinadas que já estavam bem desenvolvidas, embora sistemas de software de missão crítica sejam frequentemente bem especificados antes do desenvolvimento da aplicação. Especificações são mais importantes para interfaces externas que devem permanecer estáveis.

Arquitetura de Software

A arquitetura de um sistema de software remete a uma representação abstrata daquele sistema. Arquitetura é concernente à garantia de que o sistema de software irá ao encontro de requisitos do produto, como também assegurar que futuros requisitos possam ser atendidos. A etapa da arquitetura também direciona as interfaces entre os sistemas de software e outros produtos de software, como também com o hardware básico ou com o sistema operacional.

Implementação (ou codificação)

A transformação de um projeto para um código deve ser a parte mais evidente do trabalho da engenharia de software, mas não necessariamente a sua maior porção.

Teste

Teste de partes do software, especialmente onde tenha sido codificado por dois ou mais engenheiros trabalhando juntos, é um papel da engenharia de software.

Diversas atividades de testes são executadas a fim de se validar o produto de software, testando cada funcionalidade de cada módulo, buscando, levando em consideração a especificação feita na fase de projeto. Onde o principal resultado é o relatório de testes, que contém as informações relevantes sobre erros encontrados no sistema, e seu comportamento em vários aspectos.

Documentação

Uma importante tarefa é a documentação do projeto interno do software para propósitos de futuras manutenções e aprimoramentos. As documentações mais importantes são das interfaces externas.

Suporte e Treinamento de Software

Uma grande percentagem dos projetos de software falham pelo fato de o desenvolvedor não perceber que não importa quanto tempo a equipe de planejamento e desenvolvimento irá gastar na criação do software se ninguém da organização irá usá-lo. As pessoas ocasionalmente resistem à mudança e evitam aventurar-se em áreas pouco familiares. Então, como parte da fase de desenvolvimento, é muito importante o treinamento para os usuários de software mais entusiasmados, alternando o treinamento entre usuários neutros e usuários favoráveis ao software. Usuários irão ter muitas questões e problemas de software os quais conduzirão para a próxima fase.

Manutenção

A manutenção e melhoria de software lidam com a descoberta de novos problemas e requisitos. Ela pode tomar mais tempo que o gasto no desenvolvimento inicial do mesmo. Não somente pode ser necessário adicionar códigos que combinem com o projeto original, mas determinar como o software trabalhará em algum ponto depois da manutenção estar completa, pode requerer um significativo esforço por parte de um engenheiro de software. Cerca de ⅔ de todos os engenheiros de software trabalham com a manutenção, mas estas estatísticas podem estar enganadas. Uma pequena parte destes trabalha na correção de erros. A maioria das manutenções é para ampliar os sistemas para novas funcionalidades, as quais, de diversas formas, podem ser consideradas um novo trabalho. Analogamente, cerca de ⅔ de todos os engenheiros civis, arquitetos e construtores trabalham com manutenção de uma forma similar.

Processo de meta-modelos

O processo de desenvolvimento de software tem sido objetivo de vários padrões, que visam a certificação de empresas como possuidoras de um processo de desenvolvimento, o que garantiria certo grau de confiança aos seus contratantes.

Alguns padrões existentes atualmente:

- ISO/IEC 12207
- CMMI - Modelo de Maturidade em Capacitação - Integração (anteriormente CMM)
- ISO 9000
- ISO/IEC 15504 (anteriormente SPICE)
- MR-MPS (http://www.softex.br)
- IBM Rational Unified Process

Métodos Formais

Métodos formais em ciências da computação e engenharia de software, são técnicas baseadas em formalismos matemáticos para a especificação, desenvolvimento e verificação dos sistemas de softwares e hardwares.[1] Seu uso para o desenvolvimento de software e hardware é motivado pela expectativa de que, como em outras disciplinas de engenharia, possam contribuir para a confiabilidade e robustez de um projeto executando análises matemáticas apropriadas.[2] Entretanto, o alto custo do uso dos métodos formais faz com que, de modo geral, sejam usados apenas no desenvolvimento de sistemas de alta-integridade, nos quais há alta probabilidade de as falhas provocarem perda de vidas ou sério prejuízo.

Ver também

- Ferramenta CASE
- Gerência de projetos
- Desenvolvimento de software
- Modelos de processo de software

Referências

1. Butler, Ricky W. «What is Formal Methods?» (http://shemesh.larc.nasa.gov/fm/fm-what.html) (em inglês). Consultado em 26 março de 2015
2. Holloway, C. Michael. «Why Engineers Should Consider Formal Methods» (http://klabs.org/richcontent/verification/holloway/nasa-97-16dasc-cmh.pdf) (PDF). 16th Digital Avionics Systems Conference (27-30 outubro de 1997) (em inglês)

Butler, Ricky W. «What is Formal Methods?» (em inglês). Consultado em 26 março de 2015.

Holloway, C. Michael. «Why Engineers Should Consider Formal Methods» (PDF). 16th Digital Avionics Systems Conference (27-30 outubro de 1997) (em inglês)

Wazlawick, Raul Sidnei. Engenharia de Software: conceitos e práticas. Rio de Janeiro: Elsevier, 2013.
