TEXTO DE APOIO

AULA 2

Introdução à Engenharia de Software

Professora Kassya Christina Rigolon de Andrade

EaD | Universidade Presbiteriana Mackenzie

Sumário

PARADIGMAS E PROCESSOS DA ENGENHARIA DE SOFTWARE

É muito comum usarmos a palavra paradigma na área de computação. O que é paradigma? Paradigma é algo que serve de modelo, ou seja, um padrão. Representa uma abordagem ou filosofia em particular para construção de software. Assim como distinguimos os paradigmas da cozinha chinesa e da francesa, podemos distinguir os paradigmas do desenvolvimento orientado a objetos do paradigma procedural, que é o paradigma de programação baseado no conceito de chamadas a procedimento. Todo paradigma possui processos diferentes.

Como já vimos anteriormente, processo é um conjunto de etapas que envolvem atividades, restrições e recursos para se alcançar um objetivo ou uma saída desejada. Existem muitos processos de software diferentes, mas todos incluem quatro atividades fundamentais: especificação de software, desenvolvimento de software, validação de software e evolução de software. Um modelo de processo de software descreve um processo de uma perspectiva particular ou foca em algumas atividades. Não existe um modelo ideal e todos possuem vantagens e desvantagens.

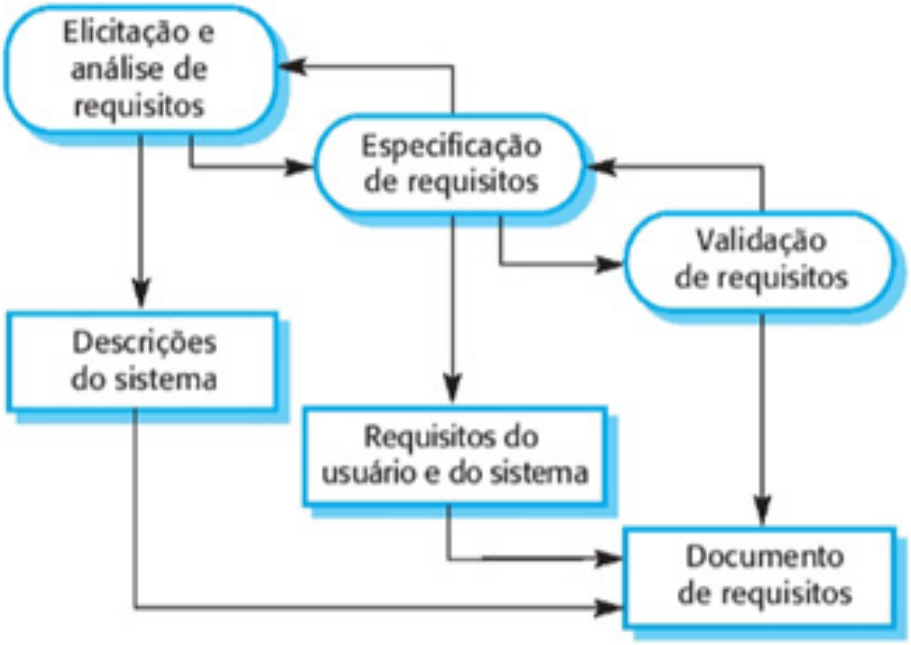
Atividades do Processo de Desenvolvimento de Software

Conheceremos, com um pouco mais de detalhes, as quatro atividades básicas do processo de desenvolvimento de software: a especificação, o desenvolvimento, a validação e a evolução.

- Especificação de software

Especificação de software ou engenharia de requisitos é o processo de compreender e definir as funcionalidades que serão necessárias para o sistema, além de identificar as restrições. É a fase mais crítica e importante do processo de desenvolvimento do software, pois os erros cometidos nessa etapa geram problemas posteriores no projeto e na implementação do sistema. A Figura 1 apresenta as atividades principais do processo de engenharia de requisitos e este é um tópico que será apresentado com mais detalhes nas próximas aulas.

Figura 1 – O processo de engenharia de requisitos

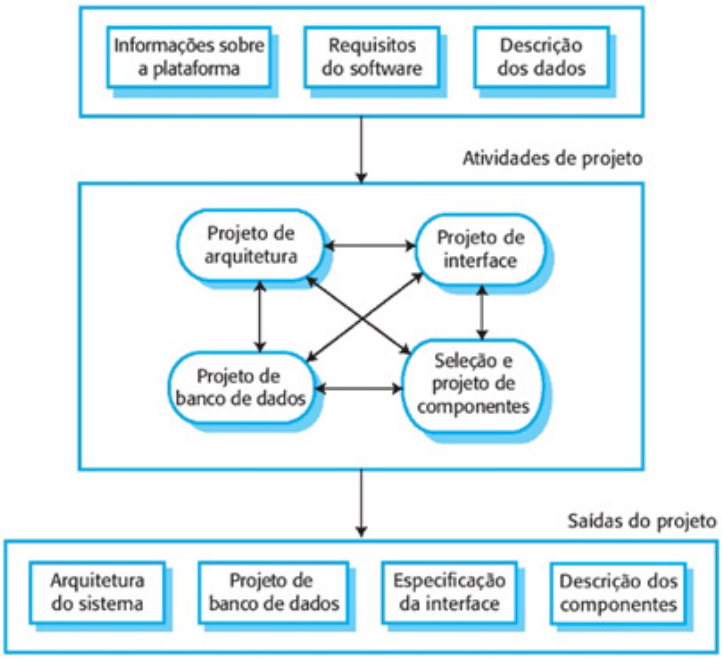


Fonte: Sommerville (2019, p. 40).

- Projeto e Implementação do software

O objetivo deste estágio é entregar um sistema executável para o cliente. Mas, para chegar nesse ponto, há várias atividades a serem realizadas antes. A Figura 2 apresenta todas as atividades do processo de projeto que acontece antes da entrega do sistema pronto.

Figura 2 – Modelo geral do processo de projeto



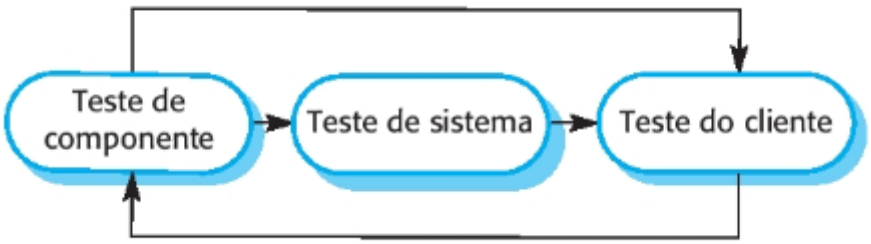
Fonte: Sommerville (2019, p. 41).

O projeto de software é uma descrição da estrutura do software a ser implementado, dos modelos e das estruturas de dados, das interfaces e dos algoritmos. Vale lembrar que, na abordagem ágil, o projeto e a implementação são intercalados, sem documentos de projeto formais produzidos durante esse processo.

- Validação do software

A validação do software envolve as atividades de verificação e validação, mais conhecidas como V&V, e objetiva mostrar que um sistema está em conformidade com sua especificação e que satisfaz as expectativas do cliente do sistema. São várias as técnicas utilizadas para verificar e validar (e você conhecerá essas técnicas mais adiante em outra disciplina do curso). A Figura 3 mostra um processo de teste com três estágios nos quais cada componente do sistema é testado individualmente e, depois, são integrados ao sistema para que ele seja testado por completo.

Figura 3 – Estágios do teste



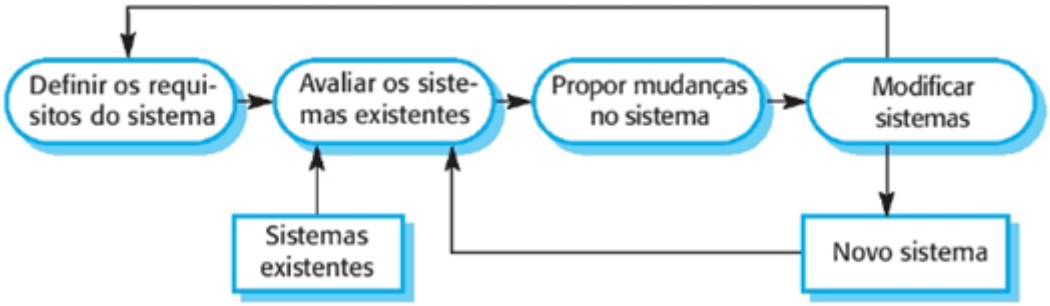
Fonte: Sommerville (2019, p. 43).

Na abordagem de desenvolvimento incremental, cada incremento deve ser testado enquanto é desenvolvido, e os testes devem ser baseados nos requisitos para aquele incremento.

- Evolução do software

Mudanças em software são muito mais baratas do que mudanças em hardware. O processo de evolução também é conhecido como manutenção do software. A flexibilidade do software é uma das principais razões que permitem que ele seja incorporado a sistemas grandes e complexos. Poucos sistemas de software são completamente novos e faz muito mais sentido visualizar o desenvolvimento e a manutenção como uma coisa só. A engenharia de software é um processo evolutivo no qual o software é alterado continuamente ao longo de sua vida útil em resposta às mudanças de requisitos e necessidades dos clientes. A Figura 4 apresenta o processo de evolução de um sistema de software.

Figura 4 – Evolução do sistema de software



Fonte: Sommerville (2019, p. 46).

Vale lembrar que essas atividades são organizadas de modo distinto nos diferentes modelos de processos de desenvolvimento que são apresentados a seguir. Para fixar ainda mais esse tópico, sugiro que assista às videoaulas da trilha de aprendizagem no ambiente virtual.

Modelos genéricos de processos de software

Os modelos de processos genéricos, aqui apresentados, também são chamados de paradigmas de processo. São descrições mais gerais e abstratas dos processos de software e podem ser utilizados para explicar as diferentes abordagens de desenvolvimento de software. Esses modelos podem ser ampliados e adaptados para criar processos de engenharia de software mais específicos. Então vamos conhecê-los:

- Modelo em cascata: representa as atividades fundamentais do processo – especificação, desenvolvimento, validação e evolução – na forma de fases de processo distintas e executadas uma após a outra.

- Desenvolvimento incremental: representa as atividades fundamentais do processo – especificação, desenvolvimento, validação e evolução, de forma intercalada. O sistema é desenvolvido como uma série de versões (incrementos), cada uma delas acrescentando funcionalidades à versão anterior.

- Integração e Configuração: baseia-se na disponibilidade de componentes ou sistemas reusáveis. O processo se concentra na configuração desses componentes reusáveis para que sejam utilizados em um novo contexto e na integração deles.

Várias tentativas têm sido realizadas para criar modelos universais a partir dos modelos genéricos. Um dos mais conhecidos é o Rational Unified Process (RUP),

desenvolvido pela empresa norte-americana Rational, é um modelo bem flexível. Foi adotado pela IBM e outras grandes empresas de software, mas não conquistou grande aceitação. O RUP foi derivado do trabalho sobre UML (Unified Modeling Language) e do Processo Unificado de Desenvolvimento de Software. Ele tenta cobrir todos os aspectos do desenvolvimento de software, está fortemente focado na documentação do sistema.

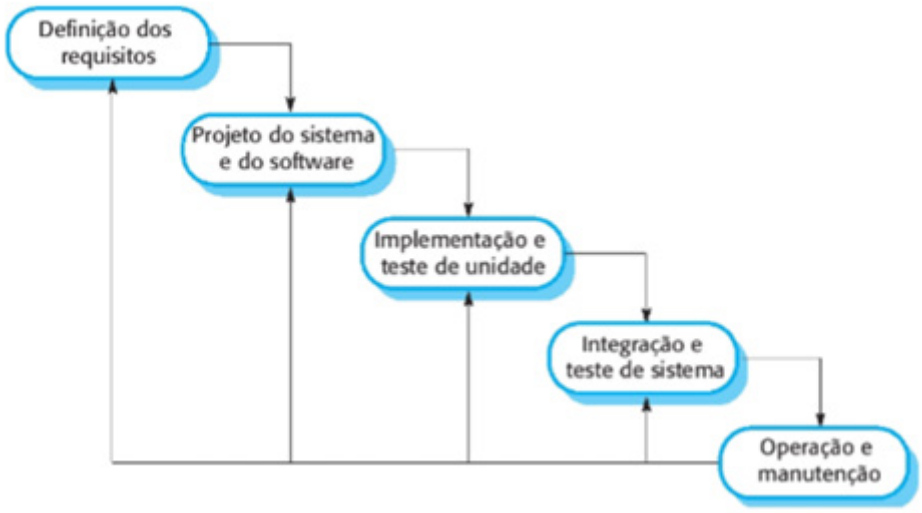
Agora, conheceremos um pouco mais a fundo os modelos de processos de software.

Modelos de processos de software

- Modelo Cascata

Foi um dos primeiros modelos propostos, as fases ou estágios são apresentados em sequência, como uma cascata. Uma fase deve estar completa para iniciar a próxima, e todas as fases envolvem a validação. A Figura 5 ilustra os estágios do modelo cascata.

Figura 5 – Estágios do modelo cascata



Fonte: Sommerville (2019, p. 33).

Os estágios do modelo cascata são exatamente as atividades fundamentais do desenvolvimento de software, e essas atividades são executados uma após a outra, sem pular nenhum dos estágios. Basicamente, os requisitos são definidos e o software é desenvolvido, sendo que cliente somente terá contato com o software no momento da entrega.

1. Definição dos Requisitos: as funcionalidades, as restrições e os objetivos do sistema são definidos por meio de levantamentos de requisitos junto aos usuários.

2. Projeto de Sistema e do software: uma arquitetura global do sistema é estabelecida a partir dos requisitos de hardware e software.

3. Implementação e teste de unidade: cada programa é testado separadamente e, depois, são feitos os testes de integração entre os programas, como um sistema completo a fim de garantir que os requisitos solicitados foram implementados. Após os testes, o sistema é entregue para o cliente.

4. Operação e manutenção: é a fase mais longa do ciclo de vida. O sistema é instalado e colocado em uso. A manutenção envolve corrigir erros não descobertos anteriormente, adicionar novas funcionalidades, atualizar o sistema de acordo com as mudanças que ocorrem no ambiente, melhorar o sistema.

O que se observa hoje em dia é que os softwares precisam ser flexíveis e precisam acomodar mudanças durante o desenvolvimento, mas o modelo cascata não se adequa a isso. Ele é mais adequado, portanto,:

- para sistemas embarcados, pois nesses sistemas o hardware é inflexível e, assim, as funcionalidades do software não sofrerão mudanças durante o desenvolvimento;

- para sistemas críticos, pois nesses sistemas os documentos de especificação e do projeto devem estar completos para que a análise seja possível. Assim, será rara alguma mudança na fase de implementação;

- para sistemas complexos que são desenvolvidos por várias empresas parceiras. Quando várias empresas estão envolvidas, é necessário que a especificação esteja completa antes de cada empresa desenvolver sua parte.

O modelo cascata não é recomendado para sistemas que permitem a comunicação informal da equipe de desenvolvimento e para sistemas em que os requisitos mudam com frequência. Para esses tipos de situações, o desenvolvimento iterativo e os métodos ágeis se adaptam melhor.

- Desenvolvimento incremental

No desenvolvimento incremental, o sistema é entregue ao cliente em incrementos (em versões) e cada incremento fornece parte da funcionalidade do sistema. Os requisitos são priorizados e aqueles com maior prioridade são incluídos nos incrementos iniciais. Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são congelados, ou seja, não são aceitas mudanças para o incremento

que está sendo desenvolvido e os requisitos para os incrementos posteriores podem continuar evoluindo e abranger requisitos já implementado. As atividades de especificação, desenvolvimento e validação são intercaladas, em vez de separadas, com feedbacks ao longo de todas as etapas, conforme ilustra a Figura 6.

Figura 6 – Estágios do desenvolvimento incremental

Fonte: Sommerville (2019, p. 35).

O desenvolvimento incremental é mais recomendado para sistemas cujos requisitos estão propensos a mudar durante o processo de desenvolvimento de software e possui três grandes vantagens:

1. O custo de implementação das mudanças nos requisitos é reduzido.

2. É mais fácil obter o feedback do cliente já que ele tem contato com cada incremento.

3. A entrega antecipada de um software útil, só com alguns incrementos prontos, é possível.

As desvantagens do Desenvolvimento Incremental são:

1. Em grandes organizações com procedimentos muito burocráticos, esse modelo pode ser incompatível por ser um processo ágil e mais informal.

2. O processo não é visível e, para os gerentes medirem o progresso, precisam de resultados regulares o que pode demandar muito tempo e custo para documentar cada versão do sistema.

3. Frequentes mudanças deixam o código bagunçado e afetam a degradação estrutural do sistema.

- Integração e configuração

É um modelo baseado em reuso sistemático, no qual sistemas são integrados a partir de componentes existentes. A abordagem orientada a reuso depende de uma grande base de componentes reutilizáveis e algum framework de integração deles. Os estágios desse modelo são: análise de componentes, modificação de requisitos, projeto de sistema com reuso e desenvolvimento e integração.

Seguem três tipos de componentes de software que são reusados com frequência:

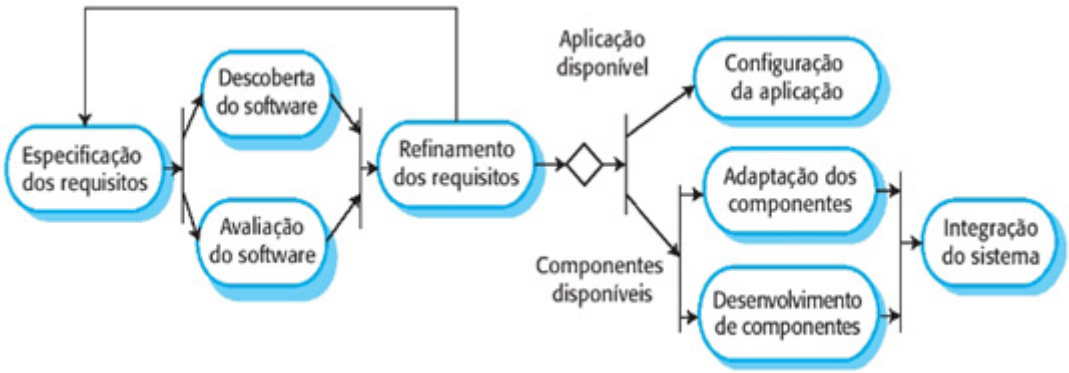
1. Sistemas de aplicação stand-alone que são configurados para utilização em um ambiente particular.

2. Coleções de objetos desenvolvidos como um componente ou um pacote a ser integrado a um framework de componentes como o Java Spring.

3. Web services desenvolvidos de acordo com os padrões de serviços.

Essa abordagem está se tornando cada vez mais usada à medida que padrões de componentes têm surgido. A Figura 7 ilustra um modelo de processo, com os estágios, para o desenvolvimento de software baseado em reúso.

Figura 7 – Estágios da engenharia de software baseado em reúso



Fonte: Sommerville (2019, p. 38).

Vamos compreender o que é realizado em cada estágio da Figura 7:

1. Especificação de requisitos: uma breve descrição dos requisitos é definida.

2. Descoberta e avaliação do software: com base na descrição dos requisitos, é feita uma pesquisa pelos componentes e sistemas que fornecem a funcionalidade

necessária. Em seguida, são avaliados para identificar qual se adequa melhor às necessidades.

3. Refinamento dos requisitos: os requisitos são refinados e definidos de acordo com os componentes escolhidos, modificações são realizadas, e a especificação do sistema é redefinida.

4. Configuração da aplicação: se o componente escolhido for uma aplicação de prateleira, ela será configurada para utilização.

5. Adaptação e integração dos componentes: se não houver aplicação de prateleira, os componentes reusáveis são modificados ou novos componentes são criados para compor a integração do sistema.

As vantagens desse modelo são a redução da quantidade de software a ser desenvolvido, reduzindo custos e riscos, o que leva, também, a uma entrega mais rápida do software; por outro lado, pode resultar em um sistema que não satisfaz às reais necessidades dos usuários. Outra desvantagem seria a perda de controle sobre a manutenção do sistema pela organização que o utiliza, uma vez que as novas versões dos componentes reusáveis não estão sobre seu controle, e sim são controladas por quem desenvolve os componentes.

Saiba Mais!

Você sabe o que é um software personalizado e um software de prateleira? Apresentarei aqui o conceito e a diferença entre eles. O software personalizado é um software desenvolvido para necessidades específicas do cliente. Ele é criado de acordo com os requisitos apontados pelo cliente. O software de prateleira, por sua vez, pode ser adquirido por qualquer empresa ou cliente que se adapte ao que é oferecido pelo programa. É um tipo de software mais genérico e generalista, cuja intenção é alcançar o maior número de clientes possível. É focado em soluções mais comuns ao mercado.

LIDANDO COM AS MUDANÇAS

As mudanças são inevitáveis em todos os projetos de software, e elas acabam elevando os custos de desenvolvimento de software, já que é um retrabalho. Apresento aqui três maneiras de lidar com as mudanças e com as variações dos requisitos as quais podem ajudar a reduzir esses custos:

1. Prototipação do sistema: com os protótipos, os usuários podem observar até que ponto o sistema os ajuda em seu trabalho, podem surgir novas ideias e podem identificar pontos fortes e fracos no software. É possível identificar vários pontos logo no início do desenvolvimento, e não somente depois que o software estiver pronto.

2. Entrega incremental: o sistema é entregue ao cliente em incrementos (em etapas) e cada incremento fornece parte da funcionalidade. Os requisitos são priorizados e aqueles com maior prioridade são incluídos nos incrementos iniciais. Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são congelados, ou seja, não são aceitas mudanças para o incremento que está sendo desenvolvido e os requisitos para os incrementos posteriores podem continuar evoluindo e abranger requisitos já implementados. A grande vantagem é que os clientes não precisam esperar até o sistema inteiro ser fornecido para usá-lo.

3. Refatoração: a melhoria da estrutura e da organização de um programa é também um importante mecanismo de suporte para mudanças.

FERRAMENTAS DE DESENVOLVIMENTO DE SOFTWARE

Para auxiliar em todas essas atividades do processo de desenvolvimento de software, temos a Engenharia de Software auxiliada por Computador (Computer-Aided Software Engineering – CASE) que é a denominação dada ao software que apoia as atividades como engenharia de requisitos, projeto, desenvolvimento de programas e testes. As ferramentas CASE incluem editores de diagramas, dicionário de dados, compiladores, debuggers, ferramentas de construção de sistemas, geradores de interfaces gráficas, entre outros. Essas ferramentas podem ser combinadas em um grande ambiente de desenvolvimento integrado chamado IDE, do inglês, Integrated Development Environment, que fornece um conjunto de recursos que podem ser utilizados tanto para a comunicação como para a operação de uma forma integrada, o que facilita muito.

DESENVOLVIMENTO ÁGIL DE SOFTWARE

Nos dias de hoje, o ambiente global muda rapidamente e as empresas precisam se adaptar a essas mudanças. Surgem novas oportunidades e mercados, as condições econômicas mudam com frequência e rapidez, assim como surgem novos produtos e serviços. O software faz parte de todas essas operações de negócios e, portanto, o software também precisa ser desenvolvido na mesma velocidade em que tudo isso acontece, para que as empresas possam ter vantagem competitiva. Nesse ambiente dinâmico, os requisitos sempre mudam, assim, há a necessidade de desenvolvimento rápido de software e de processos que possam lidar com requisitos que mudam com frequência. No final de 1990, surgiu a ideia de métodos ágeis como Programação Extrema – do inglês, Extreme Programming ou XP, o Scrum e o DSDM – do inglês, Dynamic Systems Development Method.

O desenvolvimento rápido de software passou a ser conhecido como desenvolvimento ágil, ou métodos ágeis, e se baseiam no desenvolvimento incremental.

Saiba mais!

Para conhecer um pouco mais sobre as metodologias ágeis, assista aos vídeos indicados na trilha de aprendizagem no ambiente virtual.

CONCLUINDO

Portanto, podemos observar que todos esses modelos de processos aqui apresentados propõem melhores práticas de desenvolvimento de software e precisam ir se adaptando, evoluindo e sofrendo melhorias conforme as mudanças, no ambiente onde o software vive. Essas alterações acontecem com o objetivo de aperfeiçoar cada vez mais os processos de software existentes, a fim de elevar a qualidade, diminuir os custos e reduzir o tempo de desenvolvimento de software. É um processo cíclico que envolve medição, análise e modificação dos processos.

REFERÊNCIAS

SOMMERVILLE, I. Engenharia de software. 10. ed. São Paulo: Pearson Prentice Hall, 2019.