KLS

Análise e Modelagem de Sistemas

Unidade 1

Introdução à Engenharia de Software e à Análise de Sistemas

Claudia Werlich

Imagens
Adaptadas de Shutterstock.
Todos os esforços foram empregados para localizar os detentores dos direitos autorais das imagens reproduzidas neste livro; qualquer eventual omissão será corrigida em futuras edições.

Conteúdo em websites
Os endereços de websites listados neste livro podem ser alterados ou desativados a qualquer momento pelos seus mantenedores. Sendo assim, a Editora não se responsabiliza pelo conteúdo de terceiros.

2020
Editora e Distribuidora Educacional S.A.
Avenida Paris, 675 – Parque Residencial João Piza
CEP: 86041-100 — Londrina — PR
e-mail: editora.educacional@kroton.com.br
Homepage: http://www.kroton.com.br/

Sumário

Palavras do autor

À medida que a tecnologia evolui, acaba sendo cada vez mais incorporada ao nosso estilo de vida, alguns exemplos de inovações tecnológicas encontradas no nosso dia a dia são: computação em nuvem, realidades virtual e aumentada, reconhecimento facial, entre outras tantas. Porém, não podemos nos esquecer de uma inovação importante: o software.

A união de hardware e software proporciona eficiência e agilidade para diferentes aplicações, por exemplo, não adianta ter um celular de última geração sem funcionalidades e sem aplicativos (softwares) adequados. Dessa forma, a mão de obra especializada surge como uma necessidade da demanda por softwares eficientes e inovadores, o que faz surgir, por sua vez, a necessidade de técnicas para a criação, de forma eficaz, de novos sistemas. Nesse sentido, é necessário ter conhecimentos acerca de como e quais técnicas devem ser utilizadas para a Análise e Modelagem de Sistemas.

Assim, na Unidade 1 deste livro, você verá como o desenvolvimento de software evoluiu ao longo dos anos, a importância da engenharia de software para a evolução de processos de software, bem como o desenvolvimento ágil no processo de software. Além disso, você conhecerá o papel do analista de sistemas e os processos e modelos de software aplicados no desenvolvimento deste com o objetivo de reconhecer e identificar os processos de software e suas metodologias.

Na Unidade 2, você conhecerá os conceitos de processos de negócios, modelagem e gerenciamento de ferramentas de BPM unidos à gestão de negócios e à Tecnologia da Informação, com foco nos resultados e nas melhorias dos processos de negócios. E, desta forma, serão abordados os fundamentos de processos de negócio, a modelagem de processos de negócio e o gerenciamento de processos de negócio.

Na Unidade 3, serão apresentados o processo de engenharia de requisitos, a elicitação, a especificação e a validação de requisitos e a modelagem de requisitos. Você poderá aplicar e identificar os processos de engenharia de requisitos, assim como suas técnicas e análises para sistemas. Aprenderá também algumas técnicas para especificar, modelar e documentar os requisitos de um sistema.

Na Unidade 4, você irá aplicar e praticar os fundamentos e métodos orientados a objetos; conhecerá os fundamentos da orientação a objetos, o modelo do processo unificado com suas características e fases, a introdução ao Processo Unificado da Rational (*Rational Unified Process* - RUP)

e a introdução à Linguagem de Modelagem Unificada (*Unified Modeling Language* - UML).

A utilização de técnicas para a Análise e Modelagem de Sistemas traz segurança aos desenvolvedores, pois é uma garantia de que se está desenvolvendo o solicitado pelo cliente, além de ser uma forma de estabelecer padrões de qualidade para o desenvolvimento. Possibilita, também, a redução de erros no desenvolvimento de softwares, minimiza os custos orçamentários e até o tempo de desenvolvimento se as técnicas são aplicadas de maneira adequada.

Você está entrando agora no mundo do desenvolvimento de sistemas, que é um mundo de oportunidades! Sua visão sobre um software não será mais a mesma, pois este livro trará novas ideias e oportunidades sobre o desenvolvimento de sistemas.

Fique atento às suas ideias! Quem sabe você não seja um desenvolvedor de um aplicativo inovador? Boa jornada no universo da Análise e Modelagem de Sistemas!

Unidade 1

Claudia Werlich

Introdução à engenharia de software e à análise de sistemas

Convite ao estudo

Olá! Seja bem-vindo a esta unidade do livro de Análise e Modelagem de Sistemas.

A informática está revolucionando o mundo, o que tem aumentado a busca por profissionais qualificados para desenvolver novos sistemas. Dessa forma, novas oportunidades de trabalho aparecem a todo instante, e podemos considerar que este é um momento ímpar na era digital.

Como funcionário de uma empresa de desenvolvimento de software, você está ingressando no ramo da Análise de Sistemas e, com esse cargo, muitos desafios aparecerão diariamente. Seu primeiro desafio consiste em analisar o software de um possível cliente. A empresa de desenvolvimento precisa de novas ideias para sugerir ao cliente e, quem sabe, conseguir fechar um contrato de desenvolvimento. No segundo desafio, você precisará investigar o processo de software da empresa de desenvolvimento em que você trabalha e propor melhorias nos processos de desenvolvimento de software. Após os dois primeiros desafios, você deverá avaliar e propor um modelo de processo de software a partir da observação das características do software solicitado por um novo cliente.

Para vencer esses três desafios, nesta unidade você irá aprender a reconhecer e identificar processos de softwares e suas metodologias. Na Seção 1 você aprenderá sobre os fundamentos da engenharia de software e entenderá a natureza e a evolução do software, a prática da engenharia de software, os princípios da análise de sistemas e a importância do papel do analista de sistemas no desenvolvimento de um software. Na Seção 2 serão apresentados os processos de softwares, através de uma introdução ao processo de software, da estrutura de um processo genérico de software e das tarefas e modelagem das atividades do processo de software. Na Seção 3 serão introduzidos os modelos de processos de software com a integração e validação entre as atividades do processo de software, a introdução ao conceito de modelo de processo de software, os modelos de processos prescritivo, os

modelos de processos especializados e uma introdução ao desenvolvimento ágil e seus principais métodos.

Neste momento, é aconselhável uma reflexão sobre o seu potencial: você poderá se tornar um profissional bem-sucedido na área de desenvolvimento de sistemas e ou na área da análise de sistemas. E, para isso, algumas características importantes são necessárias: busca por conhecimentos e atenção às novas tecnologias do mercado. A fim de conseguir essas características, leia com atenção esta unidade, procure entrar em sites e fóruns de desenvolvimento e fique atento aos lançamentos de novas tecnologia do mercado (tanto em hardware quanto em software).

Invista no seu potencial!

Estude, boa aula!

Seção 1

Fundamentos da engenharia de software

Diálogo aberto

Caro aluno, seja bem-vindo à seção sobre fundamentos da engenharia de software.

Trabalhar na área de engenharia de software é sempre desafiador, pois um sistema nunca é igual ao outro e há constantes novidades avindas do emprego de tecnologias diferentes. Nesse universo são necessários organização e muito empenho para produzir um software de qualidade.

Pensando nisso, você foi convidado a trabalhar como analista de sistemas em uma *software house* (empresa de desenvolvimento de sistemas) e sua primeira missão é analisar um software de um possível cliente. A empresa de desenvolvimento precisa de novas ideias para sugerir ao cliente e, quem sabe, conseguir fechar um contrato de desenvolvimento.

O cliente possui uma rede de postos de coleta de exames laboratoriais no Sul do país. Sabe-se que o sistema utilizado atualmente, um desktop sem acesso à internet, já possui quase 15 anos de uso e nunca foi atualizado. Como a empresa pensa, agora, em atualizá-lo, sua missão será descobrir: é mais viável atualizar o sistema existente ou criar um novo? Quais as novas tecnologias que podem ser utilizadas em um novo sistema para uma empresa de exames laboratoriais? Quais processos deverão ser adotados para manter o software sempre atualizado? Qual o perfil dos profissionais envolvidos no processo da análise do sistema do cliente?

O primeiro passo para responder ao questionamento acima é conhecer os fundamentos da engenharia de software, por meio do estudo desta seção que conta com: a natureza e a evolução do software, a prática da engenharia de software, os princípios da análise de sistemas e o papel do analista de sistemas.

Crie uma apresentação com as soluções encontradas para gerar um debate sobre suas ideias.

Esta seção dará uma visão inicial sobre os processos de criação de um software e você perceberá que, após conhecer esses processos, seu pensamento ficará logicamente organizado e poderá entender e planejar futuros softwares de forma sistemática. Estude esta seção e resolva a situação-problema. Desafie-se!

Bons estudos!

Não pode faltar

Na sociedade atual, a utilização de qualquer tipo de software tomou grandes proporções. Atualmente, é comum o mercado de trabalho exigir que os profissionais tenham habilidades em algum software específico, ou então que as empresas treinem seus funcionários para a utilização dele, uma vez que geralmente utiliza-se algum tipo de software para a execução de tarefas diárias.

Sommerville (2011, p. 4) estabelece que os softwares são "programas de computadores com uma documentação associada e [que] os produtos de software podem ser desenvolvidos para um determinado cliente ou para um mercado mais generalizado". Já Pressman (2016) afirma que um software de computador é um produto que profissionais da área da Tecnologia da Informação (TI) desenvolvem e para o qual darão suporte a longo prazo; além disso, abrange qualquer tipo de mídia eletrônica, consistindo na união de três elementos:

- Instruções: quando executadas, fornecem os atributos e funções de desempenhos desejados pelos usuários.
- Estruturas de dados: possibilitam aos programadores manipular as informações de forma mais adequada conforme a necessidade da aplicação.
- Documentação: é toda a informação descritiva do software, a qual detalha a operação de uso dos programas, diagramas de funcionalidades, etc.

É importante ressaltar que o software não "surgiu" do nada, houve um processo evolutivo que todos os profissionais da área de TI devem conhecer. No Quadro 1.1 pode ser observada uma cronologia da história da evolução do software e dos fatos mais impactantes que constituíram esse processo evolutivo.

Quadro 1.1 | Cronologia histórica da evolução do software

Quando?	O que?	Descrição
Década de 1940	• Programa	• O programa era executável e tinha controle total sobre o computador, sendo que o software era responsável por todo o funcionamento tanto do hardware (Sistema Operacional) quanto das operações matemáticas que teria que fazer.
Décadas de 1950 e 1960	• Sistemas Operacionais • Linguagens de programação	• Responsáveis pelo controle do hardware (realizava a interface entre o homem e a máquina). • Surgem as linguagens de programação: COBOL, LISP, ALGOL, BASIC, etc.
Décadas de 1960 e 1970	• Crise do software • Paradigmas de programação • Sistemas Operacionais mais eficientes	• Houve um grande crescimento do número de sistemas e quase não havia planejamento e controle de processos de desenvolvimento de software. Devido ao desenvolvimento informal, havia atrasos e os custos superavam o orçamento estimado (surgindo a necessidade de criar métodos de engenharia de software). • Com novas linguagens sendo criadas, nessa época o conceito de orientação a objetos é criado. • O foco, no momento, era a programação desktop, de modo que foi quando surgiram o MS-DOS da Microsoft e o Macintosh da Apple.
Década de 1980	• Computador desktop • Unix • Evolução da internet	• A década foi marcada pelo surgimento da Microsoft e pela evolução dos computadores pessoais – desktop. • O Unix (Sistemas Operacionais em rede) avançaram pelo mundo. • A internet começa a despontar como meio de comunicação.
Década de 1990	• Internet • Linux • JAVA	• A internet é amplamente utilizada. • Surge o núcleo do futuro Linux. • Nasce a linguagem JAVA, que revoluciona o paradigma da orientação a objetos.
Década de 2000	• Sistemas Operacionais gráficos • Internet	• Geração do Windows da Microsoft (Windows XP, Windows Vista, Windows 70) e versões gráficas do LINUX. • Softwares que utilizam a web como plataforma de funcionamento.
Década de 2010 e 2020	• Computação em nuvem • Aplicativos para dispositivos móveis • Inteligência Artificial	• Utilizada em larga escala por diversos seguimentos de empresas. • Com a evolução dos dispositivos móveis os softwares para aplicativos tornaram-se essenciais. • Utilização de algoritmos para a Deep Learning (Aprendizagem Profunda) e Machine Learning (Aprendizado de Máquina).

Fonte: adaptado de Fonseca Filho (2007).

Como pode ser observado no Quadro 1.1, à medida que novas formas de utilização do hardware e do software se tornam acessíveis à população, fica evidente a maior necessidade de melhorar os softwares; além disso, as necessidades dos clientes passam a mudar e vão se alterando conforme a tecnologia é disponibilizada, vejamos:

1. Na década de 1990, uma empresa precisava de um software e (talvez) de um site para que ficasse conhecida na internet; o site era utilizado como um cartão de visitas.
2. Na década de 2000, a mesma empresa precisava de um software com acesso à internet para disponibilizar recursos para seus clientes.
3. A partir de 2010, a empresa precisava de um software, um site e um aplicativo para que seus serviços pudessem ser acessados por diversos dispositivos e para que o armazenamento da base de dados estivesse na nuvem.
4. A partir de 2020, a mesma empresa já está pensando em utilizar a Inteligência Artificial para melhorar seus processos gerenciais.

Com os fatos supracitados, fica evidente que à medida que novas tecnologias se tornam populares, maiores são as necessidades da produção de software para atender as demandas da sociedade.

Assimile

Qual a diferença entre software e sistema?

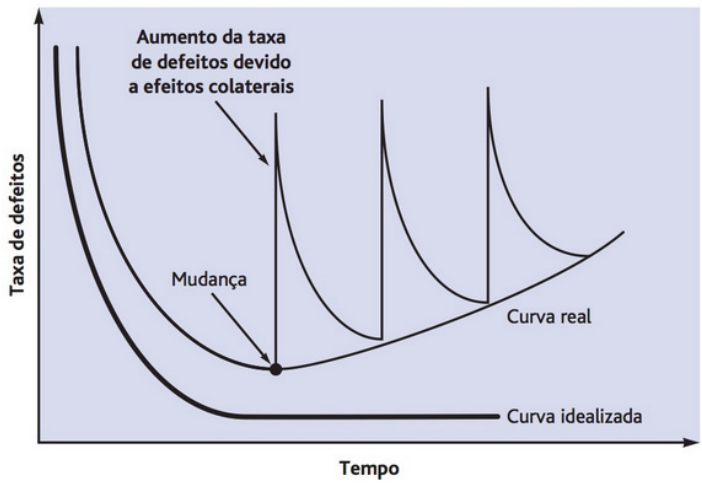
Conforme Pressman (2016), o termo software é definido como um programa de computador, uma sequência lógica de instruções a serem seguidas e/ou executadas na manipulação de um determinado conjunto de informações.

Sommerville (2011) destaca que sistema é um conjunto de componentes inter-relacionados que funcionam de forma unificada para atingir um certo objetivo. Essa definição de sistema é conhecida em algumas outras áreas, como a de engenharia de sistemas, sendo alguns exemplos: Sistema Operacional, sistema acadêmico e sistema bancário. Um sistema é formado por um determinado número de programas separados e por arquivos de configurações para eles, podendo incluir documentação específica para descrever a estrutura do sistema, documentação de usuário, etc. Um sistema possui um conjunto de partes que interagem entre si, visando a um objetivo em comum. É um conjunto de software, hardware e recursos humanos.

As mudanças sempre ocorrerão ao longo do tempo de criação e uso de um software: durante o desenvolvimento, na fase da entrega e depois de entregue. Sempre há necessidade de ajustes e correções ou ainda pode ocorrer a necessidade de incluir novas funcionalidade ao software, as quais são, muitas vezes, requisitadas pelo cliente. Além disso, o software passa por uma série de manutenções (isso acontece pois são realizadas novas solicitações do cliente) e, após realizar diversos ajustes (que podem levar a novos problemas), é possível que haja a necessidade de se criar um novo software.

Pressman (2016) destaca que, durante a vida útil de um determinado software, ele irá sofrer várias alterações, as quais podem gerar novos erros, fazendo com que a curva da taxa de defeito aumente. Esse efeito pode ser observado na Figura 1.1.

Figura 1.1 | Curva de defeitos de um software ao longo do tempo



Fonte: Pressman (2016, p. 6).

A Figura 1.1 ilustra a curva de defeitos de software (taxa de defeitos) ao longo do tempo. No gráfico, pode-se visualizar uma curva idealizada, na qual a taxa de defeitos é alta no início do desenvolvimento do software, mas que decresce com o tempo e, em seguida, estabiliza, tornando-se constante. Na prática, esse comportamento idealizado não ocorre, pois muitas vezes são necessários acréscimos de funcionalidades (a pedido do cliente) ou até mudanças no software para corrigi-lo de algum problema identificado. Nesse sentido, é apresentada a curva real, que mostra que, à medida que mudanças são introduzidas em um software, ele torna-se vulnerável, uma vez que tais mudanças podem ocasionar erros e/ou defeitos no software, como mostram

os picos no gráfico, que representam o aumento das taxas de defeitos. Muitas vezes as melhorias devem ser realizadas rapidamente e o processo da documentação é deixado de lado (para acelerar as mudanças). Todo o processo de mudança pode acarretar em um software mais frágil (com possibilidades de surgimento de mais erros).

Exemplificando

Qual a vida útil de um software? Se pensarmos na aplicabilidade de um produto, podemos dizer que ele pode se tornar obsoleto com o passar do tempo. Em se tratando de um software, sua perspectiva de usabilidade pode ser considerada longa, porém pode se tornar ultrapassado à medida que as mudanças não atendam mais as expectativas do usuário.

Pressman (2016) destaca que o software não se desgasta como ocorre com o hardware, o qual, por ser um componente físico, é suscetível e está exposto a fatores ambientais, tais como, altas temperaturas, poeira e trepidações. Todavia, softwares podem deteriorar-se com as mudanças implantadas. Importa destacar que a taxa de defeitos no hardware, além de fatores ambientais, pode ser acarretada devido a picos de processamento do software, uma vez que os recursos de hardware (memória, processador e placa de vídeo) podem não suportar a execução de um software. É o que ocorre, por exemplo, com jogos que requerem uma placa de vídeo de alto poder computacional; eles podem "travar" devido às altas temperaturas do processador e da placa de vídeo.

Sommerville (2011) destaca que as Leis de Lehman podem ser utilizadas para verificar a dinâmica da evolução dos softwares. O estudo foi realizado na década de 1980, entretanto ainda é aplicável nos dias atuais, pois as leis são genéricas e podem ser utilizadas em diferentes softwares que possuam os seguintes processos: tomada de decisão, planejamento, desenvolvimento e manutenção. As Leis de Lehman são, conforme citadas em Sommerville (2011):

- **Mudança contínua:** o software deve ser ininterruptamente adaptado, senão torna-se, aos poucos, cada vez menos eficaz.
- **Aumento da complexidade:** a cada mudança o software torna-se mais complexo. Deve-se tomar medidas para reduzir a complexidade de um software, mantendo a sua simplicidade.
- **Evolução (autorregulação):** o processo de evolução de um software é autoajustável e os atributos do sistema quase não mudam entre as versões.

- **Estabilidade organizacional:** a evolução do software tende a ser constante na sua taxa de desenvolvimento, independentemente dos recursos utilizados.
- **Conservação da familiaridade:** durante o tempo de uso do software em evolução, a taxa de mudanças tende a ser proporcional ao domínio que a equipe adquire, tornando-se constante.
- **Crescimento contínuo:** as funcionalidades do software tendem a aumentar conforme novas versões são desenvolvidas e entregues.
- **Declínio da qualidade:** caso não haja um processo de evolução do software, a qualidade deste cairá conforme os sistemas ficam desatualizados.
- **Sistema de *feedback*:** os processos de manutenção agregam sistemas de *feedback* em múltiplos níveis e *loops* para obter melhorias nos processos e no produto final.

Prática da engenharia de software

De acordo com Sommerville (2011), a prática da engenharia de software se faz necessária por dois fatores:

- Cada vez mais a sociedade (e os indivíduos que a compõem) estão dependentes dos sistemas de software e, neste contexto, devemos produzir softwares mais confiáveis e de forma mais econômica.
- A longo prazo é mais barato utilizar as técnicas da engenharia de software para evitar mudanças em algo que já tenha sido desenvolvido e principalmente que partes do software possam ser reutilizadas em outros sistemas.

Conforme Pressman (2016), a engenharia de software é uma tecnologia em quatro camadas que objetiva a disciplina, a adaptabilidade e a agilidade. As quatro camadas são:

- **Foco na qualidade:** é o objetivo final de toda a engenharia de software; toda a gestão de qualidade (das demais camadas) deve ser fundamentada em um comprometimento organizacional com a qualidade total de todas as etapas de desenvolvimento.
- **Processo:** é a base da engenharia de software; o processo é a ligação entre as demais camadas; é o que mantém as camadas de tecnologia coesas, possibilitando o desenvolvimento de forma racional (e dentro dos prazos preestabelecidos) e é a base para o controle e gerenciamento de projetos de software. É nesta camada que são produzidos

diversos artefatos (modelos, documentos, relatórios, formulários, etc.).

- **Métodos:** fornecem as informações técnicas para o desenvolvimento do software e envolvem uma grande quantidade de tarefas (comunicação, análise de requisitos, modelagem de projetos, codificação, testes e manutenção do software).
- **Ferramentas:** possibilitam um alicerce automatizado ou semi-automatizado para o processo e para os métodos; quando as ferramentas são integradas de modo que a informação criada possa ser usada por outra ferramenta, é estabelecido um suporte ao desenvolvimento de software, conhecido como engenharia de software auxiliado por computador.

A engenharia de software, segundo Sommerville (2011), preocupa-se com todos os aspectos de produção do software. Enquanto isso Pressman (2016) enfatiza que essa área engloba processos, métodos e ferramentas que possibilitam a construção de sistemas, incorporando cinco atividades específicas nesses processos: comunicação, planejamentos, modelagem, construção e entrega.

Pressman (2016) destaca sete grandes categorias de softwares e desafia a constante evolução das práticas empregadas na engenharia de software:

- **Software de sistema:** são conjuntos de programas com finalidade específica para auxiliar outros programas, por exemplo: compiladores, drivers, etc.
- **Software de aplicação:** programas independentes que têm a finalidade de resolver um problema específico de negócio, facilitando transações comerciais, decisões administrativas, e que podem ser: planilhas de cálculos, editores de textos, software exclusivo da empresa (criado e desenvolvido para a empresa).
- **Software de engenharia/científico:** programas que facilitam o entendimento e usam grandes quantidades de cálculos, geralmente utilizados em aplicações para meteorologia, astronomia, genética, etc.
- **Software embarcado:** programado de forma que só funciona para determinado produto (forno micro-ondas, geladeira, módulo de carros).
- **Software para linha de produtos:** projetado para ser utilizado por diversos clientes (com ou sem adaptações), concentrando-se em nichos de mercado, por exemplo: software contábil, software de controle de RH, etc.

- **Software de aplicações web/ aplicativos móveis:** foco em dispositivos com acesso à internet e voltados a navegadores e softwares residentes em dispositivos móveis: TV, celulares, tablets, etc.
- **Software de Inteligência Artificial:** utiliza algoritmos sofisticados para analisar e solucionar problemas complexos, inclui as seguintes áreas: robótica, reconhecimento de padrões (voz, imagens, sons), redes neurais, etc.

Um tipo de software que frequentemente é encontrado em várias instituições são os softwares legados, os quais, segundo Pressman (2016), são softwares antigos que continuam sendo utilizados, de modo que apresentam baixa qualidade, muita demora e funcionalidades que não são mais utilizadas pela empresa ou que são muito defasadas. Um software legado pode ter uma documentação deficitária ou inexistente, fazendo com que o processo de manutenção seja caro e muito demorado. O processo de evolução desse tipo de software é a criação de um novo com tecnologias mais recentes.

Assimile

A engenharia de software deve permitir que o desenvolvimento de um software seja eficiente, levando em consideração custo, qualidade e tempo de desenvolvimento. Pressman (2016) afirma que os componentes reutilizáveis devem ser projetados, adaptados e integrados em novos sistemas.

Princípios da análise de sistemas

Os princípios da análise de sistemas fundamentam-se na necessidade de realizar estudos de processos para encontrar a melhor solução para a criação de um sistema. Conforme Roth, Dennis e Wixom (2014), a análise de sistemas baseia-se em métodos e técnicas de investigação e em especificação para encontrar a melhor solução para algum problema ou necessidade computacional de determinada área de negócio, a partir das funcionalidades levantadas pelo analista de sistemas.

Em uma visão mais generalizada das fases que envolvem os processos da análise de sistema, destacam-se, conforme Pressman (2005):

- **Análise:** nesta fase são realizados estudos que objetivam a especificação do software, de modo a verificar a viabilidade (custo-benefício), definir as funcionalidades que o software deverá possuir e realizar o escopo, alocando recursos e realizando o orçamento do software. O resultado desta fase será utilizado nas próximas etapas.

- **Projeto:** nesta etapa há uma preocupação com a definição lógica do software, são elaborados os *layouts* de telas e relatórios e são criados a estrutura de banco de dados e os diagramas gráficos para o desenvolvimento do software.
- **Implementação:** nesta fase é realizada a codificação do software por meio de uma linguagem de programação (definida na fase de análise).
- **Testes:** objetivando a procura de erros, nesta fase, são realizados procedimentos de testes que verificam as funcionalidades dos itens codificados.
- **Documentação:** trata-se de documentar todos os processos (de todas as fases) e diagramas produzidos; são utilizados documentos padronizados (e personalizados por cada empresa de desenvolvimento), que servem como ferramenta de comunicação entre as pessoas envolvidas no desenvolvimento e também como parte de contrato entre as partes interessadas na produção do software.
- **Manutenção:** esta fase consiste em fazer o acompanhamento do software após ser implantado e entrar em funcionamento (durante um período), visando a registrar e corrigir falhas, propor melhorias ou incluir novas funcionalidades.

As fases apontadas por Pressman (2005), devem apresentar um processo de homologação (um aceite oficial do cliente) para validar os documentos gerados. O cliente deve estar ciente de que uma vez aprovada uma fase, caso seja alterada, haverá mudanças nos custos e implicará no dimensionamento do tempo (atrasos).

Engholm Jr. (2010) afirma que as empresas de desenvolvimento podem enfrentar diversos problemas ao não adotarem as técnicas da engenharia de software na análise de sistemas. Alguns deles são:

- Softwares com manutenção muito onerosa e demorada.
- Falta de padronização na documentação.
- Falta de controle em determinar a sequência do que deve ser realizado.
- Previsões imprecisas a respeito do prazo de entrega e do preço final do software desenvolvido.

Sommerville (2011) e Pressmann (2016) destacam alguns princípios da análise de sistemas, são eles:

- O domínio da informação de um determinado problema deve ser representado e entendido por todas as partes envolvidas.

- As funcionalidades de um software precisam ser definidas e descritas de forma genérica (inicialmente) até a forma mais genérica.
- O comportamento do software deve ser representado através das interações com o ambiente externo, ou seja, com usuários e/ou outros sistemas.
- Os diagramas que demonstram as funções e comportamentos do software devem ser divididos para revelar detalhes em forma de camadas, de modo que se decompõe um problema complexo em partes menores para facilitar sua compreensão.
- A tarefa de análise deve ir da informação essencial até os detalhes de implementação, sem a preocupação de como será realizada a codificação da solução; os detalhes sobre a implementação determinam como a solução será realizada.

O papel do analista de sistemas

O analista de sistemas possui um papel fundamental nos processos da engenharia de software conforme afirma Sommerville (2011). Esse profissional é o responsável por realizar atividades da análise de sistema como: pesquisas, planejamentos, coordenação de equipes de desenvolvimento e recomendação de alternativas de software de acordo com as necessidades de desenvolvimento ou de solução para problemas de negócios. O analista de sistemas possui como tarefas a criação, a implementação e a implantação de um software, de maneira que deve, primeiro, descobrir o que um sistema deverá fazer e depois entender e avaliar as necessidades e expectativas de cada usuário do software, a fim de que estas sejam organizadas, especificadas e documentas.

Para Elmasri e Navathe (2011), o analista de sistema desenvolve as especificações, as funcionalidades e as transações customizando as soluções para atender as solicitações dos usuários. E, neste momento, o profissional desse ramo precisa conhecer um pouco de cada área de negócio e, caso não tenha o domínio necessário sobre algum tema, deve ter a proatividade de procurar o máximo de conhecimento sobre a área que o software irá abranger.

Uma característica importante do analista de sistemas é ter uma boa visão empresarial para ajudar nos processos gerenciais da produção do software. As habilidades desejáveis para um analista de sistema são: conhecimento tecnológico atualizado, organização e método, visão gerencial, ótimo relacionamento interpessoal, entre outras.

O analista de sistemas é a "ponte" entre os programadores e os usuários finais do software, sendo ele o responsável por interpretar os anseios dos

usuários e por saber o que é ou não viável para ser desenvolvido. Cabe ao analista de sistemas colher informações com os usuários que utilizarão o software, interpretar essas informações e repassá-las aos programadores de forma técnica para que seja criado um software que atenda as expectativas do cliente e dos usuários.

Durante o desenvolvimento de um software, o analista de sistemas possui as seguintes atribuições:

- Interagir com clientes e usuários finais do software.
- Analisar custos e verificar a viabilidade do projeto.
- Fazer o levantamento das informações através de entrevistas com usuários do software.
- Levantar os dados e os requisitos do software para analisar e propor soluções.
- Criar a modelagem do software.
- Orientar os programadores, acompanhando todo o desenvolvimento do software (tanto na parte lógica quanto na parte de interface gráfica).
- Acompanhar e executar testes.
- Preparar e acompanhar toda a documentação do software.
- Gerenciar eventuais mudanças no projeto.
- Determinar padrões de desenvolvimento.
- Garantir a qualidade final do software e que este esteja de acordo com o solicitado pelo cliente.
- Realizar o monitoramento e fazer auditorias, procurando eventuais falhas.
- Planejar e aplicar treinamentos para a utilização do software desenvolvido.
- Implantar o software desenvolvido, acompanhando o processo de adaptação e integração dos sistemas do cliente.
- Proporcionar consultoria técnica para identificar as necessidades dos clientes nas mais diversas áreas de negócio.
- Pesquisar novas tecnologias, fornecedores e, se for o caso, buscar por especializações para si e para a equipe de desenvolvimento.

Reflita

Existem diversos cargos na área de TI, sendo comum a progressão deles dentro de uma empresa. Mas, será que um analista de sistemas precisa ser um ótimo programador? E um programador pode se tornar um analista de sistemas?

Nesta seção foi mostrado que um software precisa evoluir para não ficar obsoleto e foi apresentada a necessidade de incluir práticas de engenharia de software e princípios da análise de sistemas no desenvolvimento de um software. Além disso, vimos o papel do analista de sistemas em todo esse processo. Ser um analista de sistemas é ter uma carreira desafiadora, pois a cada novo projeto de Software há novas oportunidades de adquirir conhecimentos e de aperfeiçoar suas habilidades em solucionar problemas, gerenciar e coordenar equipes. Convido você a continuar seus estudos com os próximos tópicos desta unidade e a se aprofundar no fascinante mundo da criação de softwares.

Sem medo de errar

Você está trabalhando em uma *software house* e, como primeira missão nessa empresa, você precisa avaliar um software de exames laboratoriais a fim de recomendar novas tecnologias para ele, as quais, caso aprovadas, serão utilizadas. O software em questão é de uma empresa com diversas filiais no Sul do país que há 15 anos utiliza o mesmo software desktop (sem acesso à internet).

Vamos começar nossa análise avaliando se é mais viável atualizar o sistema existente ou criar um novo. O atual software que a empresa utiliza é limitado ao ambiente desktop, ou seja, não é um sistema projetado para funcionar com os recursos da internet. Esse é um importante ponto, pois, se a empresa possui filiais, é crucial que ela tenha um sistema central, o qual as filiais acessem via internet, mantendo todo o sistema integrado. Veja que, mesmo sem avaliar as funcionalidades desse sistema legado, já sabemos que não é viável usá-lo. Dessa forma, teremos que propor um novo sistema que suporte transações on-line.

Pois bem, sabendo que precisamos de um novo sistema, quais tecnologias podem ser utilizadas?

Para determinar as novas tecnologias a serem utilizadas em um novo sistema, deverá ser feito um trabalho de investigação para o qual recomenda-se os seguintes passos:

1. Visitar uma unidade da empresa para acompanhar o funcionamento do sistema.
2. Acompanhar o cadastramento da coleta de exames de um paciente a fim de observar o tempo gasto.
3. Verificar como é realizada a entrega dos resultados.
4. Descobrir qual a linguagem de programação utilizada e como é o funcionamento do banco de dados.

Após o trabalho investigativo, algumas tecnologias já podem ser apontadas:

- Criação de um site e de um aplicativo que permitam:
  - Marcar o exame.
  - Agendar a coleta em casa (caso o cliente assim deseje).
  - Acompanhar o andamento da análise laboratorial dos exames solicitados.
  - Visualizar os resultados dos exames.
  - Disponibilizar os resultados para que sejam impressos pelo paciente.
- Armazenamento do banco de dados em nuvem. Nesse caso, você deverá consultar os três grandes *players* (Amazon, Google e Microsoft) e fazer um levantamento de custos.
- Utilização da linguagem JAVA como sugestão de linguagem de programação para diminuir os custos para o cliente.

E qual deverá ser o perfil dos profissionais envolvidos no processo da análise do sistema do cliente? O ideal é ter, na equipe de desenvolvimento, analistas de sistemas com as seguintes habilidades: conhecimento tecnológico atualizado, organização e método, visão gerencial e ótimo relacionamento interpessoal. Um dos papéis do analista de sistemas é a atualização tecnológica constante, fator importante justamente nos casos em que um cliente pede ajuda para atualizar seus sistemas.

E como saber quais os processos que deverão ser adotados para manter o Software sempre atualizado? Imagine o seguinte cenário: após a entrega do software, foi verificado que nele poderiam ser utilizados códigos de barras e QR-Code. Na sua visão, o que precisa ser feito?

Acrescente itens à lista, crie uma apresentação com as soluções encontradas e as exponha aos seus colegas para gerar um debate sobre suas ideias.

Avançando na prática

Em busca de novas oportunidades de desenvolvimento

Um analista de sistemas sempre deve estar atento às novas oportunidades de desenvolvimento de novos sistemas. Muitas vezes uma empresa está "acomodada" com seus processos e é aí que esse profissional deve visualizar oportunidades de negócios para, assim, começar a empreender. Pesquise novas oportunidades de negócios; analise sites de livrarias, confeitarias, academias de ginástica, etc. Quais as vantagens que essas empresas podem ganhar com o desenvolvimento de um aplicativo? Crie uma lista de vantagens que a empresa pode ter com um aplicativo exclusivo.

Resolução da situação-problema

Para resolver essa questão, o primeiro passo é procurar por sites de empresas que podem virar aplicativos. Algumas das vantagens que ela pode ter com um aplicativo exclusivo podem ser:

- Maior visibilidade.
- Canal direto de vendas.
- Canal de relacionamento com o cliente.
- Valorização da marca com um aplicativo personalizado.
- Fidelização de seus clientes.

Agora é com você: acrescente mais vantagens à lista.

Pesquise clientes em potencial, deixe reservada uma relação de futuros clientes para que você possa oferecer consultorias como analista de sistemas.

Faça valer a pena

1. O analista de sistemas é responsável por pesquisar, desenvolver e melhorar vários tipos de softwares. Seu trabalho é analisar soluções sobre um determinado software e projetá-lo de acordo com as expectativas do cliente. Os projetos desenvolvidos podem ser: jogos eletrônicos, sites, aplicativos para celular, sistemas industriais, etc.

Analise as afirmativas a seguir sobre o papel do analista de sistemas em uma empresa de desenvolvimento de sistemas.

I. Coordena e orienta os programadores, acompanhando todo o desenvolvimento do software.

II. Gerencia eventuais mudanças no projeto, acompanhando os testes para validar as funcionalidades do software.

III. Estabelece padrões de desenvolvimento e cria os projetos lógicos do software.

IV. Coordena a implantação do software desenvolvido, acompanhando o processo de adaptação e integração dos sistemas do cliente.

Analise as alternativas a seguir e assinale a que apresenta as afirmativas corretas sobre o papel do analista de sistemas em uma empresa de desenvolvimento de sistemas.

a. Apenas as afirmativas I e II estão corretas.

b. Apenas as afirmativas I e III estão corretas.

c. Apenas as afirmativas II, III e IV estão corretas.

d. Apenas as afirmativas I, II e III estão corretas.

e. As afirmativas I, II, III e IV estão corretas.

2. A tarefa de desenvolver softwares é complexa, exigindo métodos e técnicas para investigar, levantar e solucionar problemas dos usuários. Como são muitas atividades, a análise de sistemas ajuda com processos específicos (fases de desenvolvimento) para tornar as atividades mais gerenciáveis para o analista de sistemas.

Analise as afirmativas a seguir sobre das fases que envolvem os processos da análise de sistema.

I. Projeto: nesta etapa há uma preocupação com a definição lógica do software; são elaborados os *layouts* de telas e relatórios; há a criação da estrutura de banco de dados e dos diagramas gráficos para o desenvolvimento do software.

II. Análise: nesta fase são realizados estudos que objetivam a especificação do software, verificando a viabilidade (custo-benefício) e definindo as funcionalidades que o software deverá possuir.

III. Documentação: trata-se de documentar todos os processos (todas as fases); serve como ferramenta de comunicação entre as pessoas

envolvidas no desenvolvimento e também é utilizada como parte de contrato entre as partes interessadas na produção do software.

IV. Manutenção: nesta fase são realizados a codificação e os testes no código fonte do software, utilizando técnicas de refinamento e buscas unitárias para estabelecer a precisão lógica dos componentes do software.

V. Implementação: esta fase consiste em fazer o acompanhamento do software após este ser implantado e entrar em funcionamento (durante um período), visando a registrar e corrigir falhas, propor melhorias ou incluir novas funcionalidades.

Analise as alternativas a seguir e assinale a que apresenta as afirmativas corretas sobre o papel do analista de sistema em uma empresa de desenvolvimento de sistemas.

a. Apenas as afirmativas I, II, IV e V estão corretas.

b. Apenas as afirmativas I, II e III estão corretas.

c. Apenas as afirmativas II, III e V estão corretas.

d. Apenas as afirmativas I, II e IV estão corretas.

e. As afirmativas I, II, III, IV e V estão corretas.

3. Pressmann (2016) enfatiza que um software legado é um software antigo que continua sendo utilizado, porém apresenta baixa qualidade, demora e funcionalidades que não são mais utilizadas pela empresa ou que estão muito defasadas.

Assinale a alternativa correta sobre um software legado.

a. Um software legado pode ter o seu código fonte totalmente reaproveitado, facilitando o processo de atualização para um software novo.

b. Um software legado pode ter o seu código fonte parcialmente reaproveitado, facilitando o processo de atualização para um software novo.

c. Um software legado possui a vantagem de ser atualizado com custos relativamente baixos em vista de um software novo.

d. Um software legado deve ser mantido na empresa mesmo após a implantação de um novo software, pois não é possível um novo software substituir um software legado.

e. Um software legado pode ter uma documentação deficitária ou inexistente, fazendo com que o processo de manutenção seja caro e muito demorado.

Seção 2

O processo de software

Diálogo aberto

Olá, seja bem-vindo!

O cotidiano de um Analista de Sistemas é bem agitado, são muitas tarefas que devem ser executadas: visita ao cliente, orientação aos programadores, pesquisas e, claro, muito planejamento. A organização do tempo deverá ser um exercício constante na vida profissional do Analista de Sistemas, caso contrário muitas tarefas ficarão esquecidas e, mais tarde (durante o desenvolvimento do software), esse esquecimento pode gerar uma série de prejuízos. Pensando neste aspecto, você receberá um desafio.

Como Analista de Sistemas em uma Software House, você realiza diversas tarefas, atendendo diferentes clientes. Alguns clientes têm relatado um incômodo, quanto ao tempo de entrega do produto final, o software. Para ajudar a empresa, você possui a missão de investigar o Processo de software da empresa e propor melhorias. Para isso, deverá se concentrar em responder os seguintes questionamentos: Quais as atividades principais de um Processo de Software? Como você poderia melhorar o Processo de construção de um software?

Para ajudar, nesta seção estaremos abordando as atividades do Processo de Software, que é uma série de atividades que estão relacionadas, tendo como resultado um produto (o software). Estaremos apresentando a Introdução ao Processo de Software, a Estrutura de um Processo Genérico de Software, as Tarefas e Modelagem das Atividades do Processo de Software e a

Integração e Validação entre as Atividades do Processo de Software.

Boa aula!

Não pode faltar

Um Processo de Software, conforme destaca Sommerville (2011), é um conjunto de atividades e resultados que estão relacionados, que levam à produção e ao resultado de um software desenvolvido. Pressman (2016) enfatiza que na Engenharia de Software um processo não é uma determinação rigorosa sobre como ele deve ser desenvolvido, ao contrário, é uma abordagem adaptável que possibilita à equipe de desenvolvimento escolher os processos que melhor se enquadram na filosofia da empresa (de

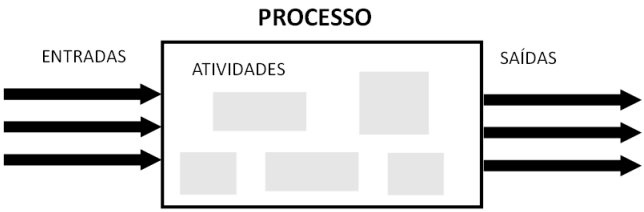
desenvolvimento) com o foco na qualidade do produto, no prazo de entrega e na redução de custos.

Sommerville (2011) afirma que a abordagem sistemática usada pela Engenharia de Software para produção de software é chamada Processo de Software. Um Processo de Software pode conter diversas atividades que normalmente são: especificação, projeto, implementação, validação, manutenção e evolução. Engholm Jr. (2010) relaciona os seguintes aspectos sobre o Processo de Software:

- Cria uma padronização na forma de gerar os serviços e produtos.
- Permite que sejam repetidos os serviços e produtos, reutilizando partes já produzidas e padronizadas.
- Retém o conhecimento na empresa, permitido que novos integrantes possam dar continuidade aos processos definidos previamente.
- Serve para definir e guiar as atividades de um Projeto de Software.
- Permite a especificação de todo o processo para o desenvolvimento do software (ou partes dele).
- Determina as tarefas que deverão ser executadas para a equipe e de forma individual.
- Reduz riscos, tornando os resultados mais previsíveis.
- Proporciona visões comuns para a equipe de desenvolvimento, facilitando a comunicação.
- Pode ser utilizado como um *template*, sendo utilizado em outros projetos, permitindo agilidade em novos projetos de software.

Na definição dos Processos de Software, segundo Engholm Jr. (2010), são definidas uma série de parâmetros: (i) evento que determina o início do processo; (ii) matriz de responsabilidades; (iii) atividades que serão executadas e suas sequencialidades; (iv) entradas e saídas de cada atividade; (v) regras e políticas a serem aplicadas na realização das atividades; (vi) infraestrutura necessária; e (vii) resultado gerado na execução de cada processo. A Figura 1.2 demonstra graficamente que, em cada Processo de Software, podem ocorrer diversas atividades (sequenciais ou em paralelo).

Figura 1.2 | Representação de um Processo de Software

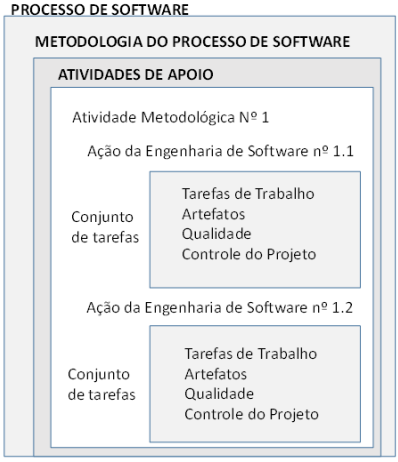


Fonte: adaptada de Engholm Jr. (2010, p.43).

Conforme a Figura 1.2, um Processo de Software possui inúmeras entradas e saídas. O processo se constitui em uma série de atividades que serão executadas de forma padronizada, agrupadas em fases (essas atividades mudam conforme há a troca de fase), sendo que, em cada fase, serão definidos: as responsabilidades (quem fará o quê), prazos de entrega e como o objetivo será alcançado. A Figura 1.3 ilustra um esquema para o processo de software. Podemos observar que:

> cada atividade metodológica é composta por um conjunto de ações de engenharia de software. Cada ação é definida por um conjunto de tarefas, o qual identifica as tarefas de trabalho a ser completadas, os artefatos de software que serão produzidos, os fatores de garantia da qualidade que serão exigidos e os marcos utilizados para indicar progresso. (PRESSMAN, p. 31, 2016)

Figura 1.3 | Exemplo de Metodologia de Processo de Software



Fonte: adaptada de Pressman (2016, p.32).

Assimile

Um problema que atinge o desenvolvimento de um software é a troca de pessoal que naturalmente ocorre em qualquer situação. Engholm Jr. (2010) afirma que, estando os Processos de Software bem definidos, os conhecimentos produzidos em cada etapa (do Processo) estarão preservados, garantindo a sua continuidade.

O Processo de Software adotado em uma empresa pode ser completamente diferente de outra empresa, cada qual procura encontrar e estabelecer atividades que visam aumentar a qualidade e baixar o custo de produção do software produzido. Independente do modelo de Processos de Software adotado pela empresa de desenvolvimento, todos utilizam uma Estrutura de Processo Genérico de Software, com atividades pré-estabelecidas.

As atividades de um determinado Processo de Software constituem um conjunto mínimo para se obter um produto de software (o software finalizado e entregue ao cliente). Em um Processo Genérico de Software, os processos podem ser diferentes, mas podemos identificar quatro atividades fundamentais em toda a produção de software, conforme Sommerville (2011):

- **Especificação de software**: definição do que será desenvolvido, suas restrições e funcionalidades.
- **Projeto e Implementação de software**: projeto e desenvolvimento (programação) do software atendendo às especificações.
- **Validação de software**: verificação se o que foi construído atende as solicitações do cliente.
- **Evolução de software**: evolução do software para que acompanhe as alterações solicitadas pelo cliente.

Cada atividade do Processo Genérico de Software é composta por um conjunto de atividades da Engenharia de Software. Pressman (2016) afirma que uma metodologia genérica da Engenharia de Software é composta de cinco atividades, que são:

- **Comunicação**: com a intenção de entender os objetivos do projeto, a comunicação entre os envolvidos é a primeira ação primordial, para entender os requisitos (as funcionalidades) do software a ser realizado.
- **Planejamento**: é realizado um "mapa", um plano de projeto do software a ser realizado, descrevendo as tarefas técnicas, os riscos, os recursos, os produtos resultantes e um cronograma de trabalho (para acompanhar o desenvolvimento do software).

- **Modelagem**: são criados modelos (diagramas) para melhor entendimento das necessidades do software, os modelos são utilizados para realizar a codificação do software e para validação das partes envolvidas no projeto.
- **Construção**: realização da codificação (baseada nos modelos criados anteriormente), nesta fase também são realizados os testes para validar os códigos de programação gerados.
- **Entrega**: o software é entregue parcialmente ou na sua totalidade, onde o cliente realiza testes e fornece um *feedback*; nesta fase são realizadas adaptações e correções no software por um determinado período (acordado entre as partes).

Pressman (2016) destaca que as atividades metodológicas (citadas anteriormente) devem ter uma série de tarefas que darão suporte no acompanhamento e controle do projeto, controlando os riscos, fazendo revisões técnicas, etc.. Entretanto, a sequencialidade ou não de cada atividade genérica pode ser dividida de acordo com o **Fluxo de Processo**. Segundo Pressman (2016), esse fluxo é usado para descrever como as atividades metodológicas de cada Processo são organizadas. Os Fluxos de Processos podem ser:

- **Fluxo de Processo Linear**: caracteriza-se por realizar em forma sequencial as cinco atividades metodológicas apontadas por Pressman (2016), inicializando pela Comunicação e terminando na fase da Entrega.
- **Fluxo de Processo Interativo**: possui como característica a repetição de uma ou mais atividades antes de avançar para a próxima atividade.
- **Fluxo de Processo Evolucionário**: as atividades são executadas de modo circular, cada ciclo envolve as cinco atividades, gerando uma versão mais completa (é um processo incremental).
- **Fluxo de Processo Paralelo**: as atividades são realizadas de forma paralela, onde duas ou mais atividades podem ser executadas simultaneamente, por exemplo, a atividade de Comunicação pode ocorrer em paralelo com a atividade de Análise.

Reflita

Os Fluxos de Processo de um Software conduzem a execução do software. Será que uma empresa desenvolvedora de software pode usar Fluxos de Processos diferentes para realizar o mesmo tipo de software, mas com clientes distintos?

O desenvolvimento de um software requer muito planejamento e um software nunca é igual ao outro. Uma universidade pode solicitar um software de controle Acadêmico, caso outra universidade solicite o mesmo tipo de software, dificilmente (para não afirmar quase que impossível) os dois softwares serão iguais. Pressman (2016) afirma que projetos diferentes exigem conjuntos de Tarefas e Modelagem das Atividades do Processo de Software diferenciados. Os Analistas de Sistemas determinam o conjunto de tarefas baseados nos problemas e nas características do projeto que será executado.

Um conjunto de tarefas determina o que deverá ser feito para alcançar os objetivos de uma determinada ação, dentro do Processo de Software. Um exemplo é a codificação das telas de um Sistema, o programador recebe um conjunto de requisitos que as telas deverão possuir, após a conclusão da programação, o trabalho do programador será avaliado conforme as especificações que ele recebeu para fazer as telas do Sistema.

Exemplificando

Observe um conjunto de tarefas (atividades) na fase de Planejamento de um Software:

ATIVIDADES DE PLANEJAMENTO DE UM SOFTWARE		
FASE	**ATIVIDADES**	**RESULTADOS**
• Planejamento	• Levantamento de Requisitos. • Especificação dos Requisitos. • Estimativas de Prazos. • Estimativa de Recursos.	• Documentação do Levantamento de Requisitos. • Documentação da Especificação de Requisitos. • Plano de ação para determinar os prazos. • Alocação de Recursos para criação do software.

Conforme Sommerville (2011) o Modelo de Processo de Software é uma descrição simplificada do Processo que especifica as atividades para o desenvolvimento, define os produtos de cada atividade, determina os papéis dos envolvidos no desenvolvimento, oferecendo um roteiro para a Engenharia de Software.

A existência de um Processo de Software não garante a qualidade do software e muito menos que o software será entregue no prazo combinado, também não é certo que as funcionalidades do software estarão de acordo com o que o cliente solicitou. A qualidade do software produzido é diretamente influenciada pelos padrões de qualidade impostos durante os Processos de software (durante a produção do software) sendo necessário estabelecer

procedimentos e padrões para garantir a qualidade dos Processos. Pfleeger (2004) destaca que uma das vantagens de fazer a Modelagem dos Processos é a possibilidade de examinar e procurar meios de aprimoramento antes de finalizar o produto (o software produzido).

O Processo de Software deve ser avaliado para certificar que ele atenda a um conjunto de critérios básicos. O Processo de Software pode passar por uma série de critérios pré-estabelecidos que ajudam a garantir a Integração e Validação entre as Atividades do Processo de Software. Pressman (2016) destaca uma série de abordagens de avaliação e aperfeiçoamento dos Processos de Software: (i) SCAMPI, (ii) CBA IPI, (iii) SPICE e (iv) ISO 9001:2000 para software.

A abordagem SCAMPI (Método Padrão CMMI de Avaliação para Aperfeiçoamento de Processos da CMMI - *Standard CMMI Appraisal Method for Process Improvement*) fornece um modelo de avaliação do Processo em cinco etapas – início, diagnóstico, estabelecimento, atualização e aprendizado. Define regras para assegurar a objetividade na classificação das avaliações, e ainda:

- Ajuda a coletar e reunir evidências por meio de apresentações, documentos e entrevistas.
- Transforma em anotações todas as evidências do que foi observado.
- Converte as anotações em declarações de acertos ou falhas quando comparadas às práticas do CMMI.
- Converte as declarações em descobertas preliminares.
- Obtém a validação das descobertas preliminares e transforma as descobertas preliminares em definitivas.

Assimile

CMMI (Modelo de Capacidade e Maturidade Integrado - *Capability Maturity Model Integration)* é um conjunto de práticas que orientam a implementação de uma série de atividades com o objetivo de alcançar uma meta pré-estabelecida, aumentando o amadurecimento organizacional e auxiliando a obter os resultados esperados pela área de TI. Existem três modelos diferentes do CMMI:

- CMMI *for Development*: apresenta melhores práticas para desenvolver produtos e serviços.
- CMMI *for Acquisition:* mostra as melhores práticas para adquirir produtos e serviços.
- CMMI *for Services:* indica as melhores práticas para entregar serviços.

Segundo Pressman (2016), a abordagem CBA IPI (CMM *Based Appraisal for Internal Process Improvement* - Avaliação para o Aperfeiçoamento do Processo Interno baseado na CMM (*Capability Maturity Model* - Modelo de Maturidade em Capacitação), fornece uma técnica de diagnóstico para avaliar a maturidade relativa de uma organização de software. O método possui a capacidade de identificar pontos fortes e fracos dos processos e viabiliza a possibilidade de priorização das melhorias mais relevantes.

Pressman (2016) afirma que a abordagem SPICE (ISO/IEC15504) é um padrão que define um conjunto de requisitos para avaliação do Processo de Software, possui a finalidade de auxiliar as organizações no desenvolvimento de uma avaliação objetiva da eficácia de um processo de qualquer software. Este método fornece uma estrutura para a avaliação de Processos de Software e esta estrutura pode ser empregada nas organizações envolvidas na produção de um software.

A abordagem ISO 9001:2000 para software, conforme Pressman (2016), é um padrão genérico aplicável a qualquer organização que precise aplicar um padrão global de qualidade em seus produtos, sistemas ou serviços. Existem vários modelos de referências para ajudar a garantir a qualidade e que são aplicáveis a um software, alguns destes modelos são: ISO/IEC 9126, a ISO9000, a ISO9001, a ISO/IEC12207. A ISO9001 descreve um modelo de garantia de qualidade em projetos, instalações, desenvolvimento e assistência técnica. Esta norma pode ser aplicada especificamente na área de desenvolvimento, fornecimento e manutenção de software por meio dos roteiros descritos na norma ISO 9000-3. A ISO/IEC 9126 descreve as características de um software de qualidade.

Os erros que ocorrem durante o Processo de Software podem ser controlados utilizando uma abordagem metodológica. O Analista de Sistemas deve estar atento ao surgimento de novas metodologias, testá-las e, se forem apropriadas, utilizar durante o Processo de Software. O objetivo é criar um software com qualidade com o mínimo de erros e com a aprovação do cliente. Levando em consideração os preceitos apresentados nesta seção, convido você a conhecer os Modelos de Processos de Software que serão apresentados na próxima seção, dando continuidade ao Processo de Software.

Sem medo de errar

Chegou o momento de resolver o desafio que lhe foi proposto. O desenvolvimento de um software requer a mobilização de diversos profissionais, cada um contribuindo com sua função. Você, como Analista de Sistemas em uma Software House, recebeu como missão rever o processo de Software da

empresa e propor melhorias. Podemos começar essa investigação, fazendo um levantamento das principais atividades de um Processo de Software.

Geralmente as atividades de um Processo de Software estão divididas em: Especificação (Análise), Projeto, Implementação, Validação, Manutenção e Evolução.

Para exemplificar as atividades de um Processo de Software, veja algumas atividades:

- Na Especificação: são realizados o estudo de Viabilidade, a Elicitação de Requisitos, a Especificação de Requisitos e a Validação dos Requisitos.
- Projeto: são definidas as estruturas modulares do software, as *interfaces* gráficas e as estruturas de dados (do banco de dados).
- Implementação: tudo o que foi decidido nas atividades de Especificação e Projeto é passado para uma linguagem de programação e o banco de dados é criado.
- Validação: São realizados testes para validar tanto os códigos dos programas quanto a verificação dos requisitos (se o software atendeu aos requisitos impostos pelo cliente).
- Manutenção e Evolução: são atividades contínuas a fim de melhoramento do software e inclusão de novos recursos.

É comum às atividades de Especificação que envolvem todo o processo de busca de informação sobre o software a ser construído funcionar em paralelo às atividades de Projeto (assim que as funcionalidades do software forem levantadas, já é realizado o Projeto do Software), e como não podemos deixar os programadores desocupados, as atividades da Implementação (codificando o software) caminham em paralelo às atividades de Projeto (desde que aprovadas).

Agora que já sabemos quais as principais atividades, podemos pensar em como melhorar o processo de construção de um software. Para isso, pensando na qualidade final dos Processos de Software, é necessário estabelecer uma série de critérios de validação das atividades do Processo de Software, implantando um (ou mais) métodos de abordagem de avaliação e melhoria dos processos, podendo ser: SCAMPI, CBA IPI, SPICE e ISO 9001:2000.

Entretanto, você deverá fazer algumas pesquisas para ajudar a sua empresa. Pesquise na Internet:

1. Normas específicas da ISO 9001:2000. Faça um relatório indicando quais itens dessa norma podem ser adotados no Processo de Software.
2. Novas metodologias ágeis que podem acelerar o desenvolvimento.

Crie uma apresentação em slide para que possa mostrar os resultados para seus colegas.

Lembre-se que a pesquisa é um fator importante na carreira do Analista de Sistemas.

Bom trabalho!

Faça valer a pena

1. Pressman (2016) afirma que um Processo de desenvolvimento de software é um conjunto de atividades, ordenadas (de forma parcial), com a intenção de obter um produto de software. É considerado um dos principais mecanismos para se obter software de qualidade e cumprir os contratos de desenvolvimento firmados com o cliente.

Sobre os Processos de Software, analise as afirmativas a seguir.

I. Os Processos de Software permitem que sejam repetidos os serviços e produtos, reutilizando partes já produzidas e padronizadas.

II. Os Processos de Software servem para definir e guiar as atividades de um Projeto de Software.

III. Os Processos de Software proporcionam visões comuns para a equipe de desenvolvimento, facilitando a comunicação.

IV. Os Processos de Software determinam as tarefas que deverão ser executadas pelas equipes e de forma individualizada.

Analisando as afirmativas apresentadas, é correto o que se afirma em:

a. Apenas II e III.

b. Apenas I e III.

c. Apenas I e II.

d. Apenas III.

e. I, II, III e IV.

2. Sommerville (2011) descreve que a utilização de um Processo de Software é apontada como um fator essencial para o sucesso de empresas de desenvolvimento de software, ajudando no gerenciamento dos Processos de desenvolvimento.

Um Processo de Software pode ser entendido como:

a. Uma tarefa específica utilizada no desenvolvimento de um software.

b. Um conjunto estruturado de atividades exigidas para desenvolver um software.

c. Uma metodologia de controle de agilidade, utilizada no desenvolvimento do software.

d. Um diagrama utilizado para especificar o Fluxo de Processos no desenvolvimento do software.

e. Um padrão de codificação, amplamente utilizado para reutilizar o código do software.

3. Pressman (2016) afirma que a Comunicação é um item fundamental no Processo de Software. Outros autores levam em consideração que a Comunicação está implícita em todos os Processos de desenvolvimento, pois existe a necessidade evidente de comunicação entre as partes envolvidas, começando no levantamento dos requisitos até a entrega do produto final.

Assinale a alternativa correta que apresenta as principais atividades que normalmente um Processo de Software possui.

a. Especificação, implementação, validação, manutenção e evolução.

b. Projeto, implementação, codificação, validação, manutenção e evolução.

c. Implementação, validação, projeto, construção, manutenção e evolução.

d. Especificação, projeto, implementação, validação, manutenção e evolução.

e. Evolução, projeto, implementação, codificação, validação, manutenção e evolução.

Seção 3

Modelos de processos de software

Diálogo aberto

Olá, seja bem-vindo!

O uso de smartphones vem revolucionando a maneira das pessoas utilizarem Softwares. Existe APP para pagar contas, controlar gastos, localizar lugares e pessoas, e uma infinidade de outras utilidades. Nesse universo, dois grandes Sistemas Operacionais se destacam, o Android e o iOS. Você sabia que para desenvolver um Software (APP) nativo para esses dois sistemas é preciso utilizar diferentes linguagens de programação? Ou seja, para um mesmo APP nativo, são necessários dois projetos de desenvolvimento e duas equipes. Com tanto trabalho, como isso pode ser feito de maneira organizada, controlada e eficiente? Só há uma maneira! Usando um Modelo de Processo de Software que seja adequado ao projeto, à equipe e às expectativas do cliente.

Nesta seção serão apresentados vários Modelos de Processos de Software, disponíveis no mercado, sendo que alguns são bem antigos, mas, mesmo assim, são utilizados em vários projetos. Na verdade, tudo depende do tipo de Software a ser produzido, combinado com as expectativas do cliente. Todos os modelos possuem a finalidade de evitar o caos no desenvolvimento e estabelecer um Fluxo de Trabalho.

Você trabalha como Analista de Sistemas, em uma *software house* e recebeu um novo cliente, que deseja fazer um Software para sua loja de brinquedos *online*. O Software deverá permitir que os usuários comprem produtos da loja. Além do *site,* é necessário fazer um aplicativo, caso o usuário queira acessar a loja para comprar ou alugar brinquedos (o aluguel somente estará disponível na versão em aplicativo). Você deverá determinar: qual Modelo de Processo de Software será mais indicado para o mais novo projeto de Software?

Para realizar essa missão, teremos uma introdução ao conceito de Modelo de Processo de Software. Veremos os Modelos de Processos Prescritivos e os Modelos de Processos Especializados, além de uma introdução ao desenvolvimento ágil e seus principais métodos.

Se for necessário, pesquise mais sobre cada item, procurando imagens sobre cada Modelo de Processo.

Crie uma apresentação para compartilhar com seus colegas e trocar ideias sobre os Modelos de Processos de Software.

Boa jornada de estudos!

Não pode faltar

Um Processo de Software é constituído por várias atividades, cuja finalidade é ter como resultado um Produto de Software. Durante a fase de desenvolvimento de um Software existem muitas tarefas, que devem ser distribuídas entre os membros da equipe. Sommerville (2011) destaca que dentre as atividades do Processo de Software estão: o estudo da viabilidade, a análise dos requisitos, a especificação, a arquitetura de software, implementação (codificação), testes, documentação, suporte e treinamento e manutenção.

Um Processo de Software não precisa ter obrigatoriamente as atividades listadas por Sommerville (2011), além disso essas atividades podem ou não estar ordenadas, como afirma Pressman (2016). Existem atividades genéricas e que aparecem na maioria dos Processos de Software e que são apontadas por Sommerville (2011):

Análise e Especificação: são realizadas as definições sobre o Software (a ser produzido) e determinados seus requisitos (funcionalidades) e suas restrições.

Projeto: é realizada a alocação de recursos (Hardware e Software) e são identificadas e definidas as abstrações do funcionamento do Software.

Implementação e Teste Unitário: é todo o processo de codificação do Software (seu desenvolvimento realizado por Analistas e Programadores), é a fabricação do Software.

Integração e Verificação: é todo o processo de avaliação, correção e validação, considerando a qualidade final do Software.

Operação e Manutenção: nesta etapa são considerados todos os processos de alterações realizadas no Software (após ele estar em funcionamento).

Para o gerenciamento das atividades de Processo de Software são utilizados os **Modelos de Processos de Software**. Um Modelo de Processo de Software tem como objetivo propiciar estabilidade, controle e organização das atividades e é uma representação (geralmente gráfica) dos objetos e atividades envolvidas no Processo de Software. Pfleeger (2004) afirma que existem muitas técnicas e ferramentas para a Modelagem de Processos e não há um consenso em determinar o melhor modelo. Cada empresa adota seu Modelo de Processo de Software, realizando adaptações a cada Software produzido.

Pressman (2016) destaca que um Modelo de Processo de Software é um guia exclusivo para as atividades da Engenharia de Software, definindo um fluxo de todas as atividades, ações e tarefas, o nível de interação entre as atividades, os artefatos que serão produzidos e a organização do trabalho que deve ser realizado. Existem diversos Modelos de Processos de Softwares que possuem características diferentes. Destacam-se os seguintes: Modelos de Processos Prescritivos, Modelos de Processos Especializados e Modelos de Desenvolvimento Ágil.

Modelo de Processo Prescritivo

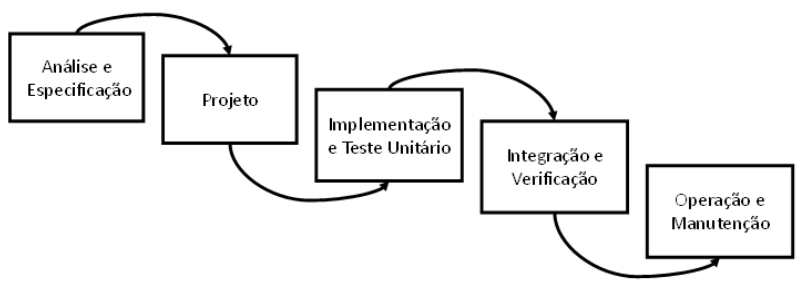
O Modelo de Processo Prescritivo (conhecido também como Modelos de Processos Tradicionais) consiste em um conjunto de elementos do processo, que podem ser ações de engenharia de software, produtos de trabalho e mecanismos que garantem a qualidade e o controle de mudanças nos projetos de desenvolvimento de um sistema de software (Pressman, 2016). Esse tipo de modelo prescreve os relacionamentos, ou seja, como os elementos dos processos são interligados, com a finalidade de estruturar e ordenar o desenvolvimento de um Software.

Conforme Pressman (2016), as tarefas ocorrem de forma sequencial, com diretrizes bem definidas, uma vez que indicam uma série de atividades metodológicas, ações, tarefas, artefatos, garantias de qualidade e mecanismos de controle de mudanças para cada projeto. Para cada Modelo de Processo Prescritivo também é indicado um Fluxo de Processo (ou Fluxo de Trabalho) ou seja, como esse conjunto de elementos do processo está interligado.

Alguns dos **Modelos de Processos Prescritivos** são os seguintes: **Modelo Cascata**, **Modelo Incremental**, **Modelos Evolucionários** (divididos em: Prototipação e Espiral) e **Modelos Concorrentes**.

O Modelo Cascata (conhecido como Ciclo de Vida Clássico de um Sistema ou abordagem *Top-down*) possui enfoque sistemático e sequencial dos Processos, cada fase é iniciada somente após a conclusão da fase anterior. A Figura 1.4 representa o Modelo Cascata.

Figura 1.4 | Modelo de Processo Prescritivo: Modelo Cascata



Fonte: elaborada pela autora (2020).

O Modelo Cascata, segundo Pressman (2016) é um dos mais antigos e ainda utilizados na Engenharia de Software. A Figura 1.4 destaca os estágios do Modelo Cascata com as cinco atividades fundamentais deste modelo. Vejamos a definição dessas cinco etapas segundo Sommerville (2011):

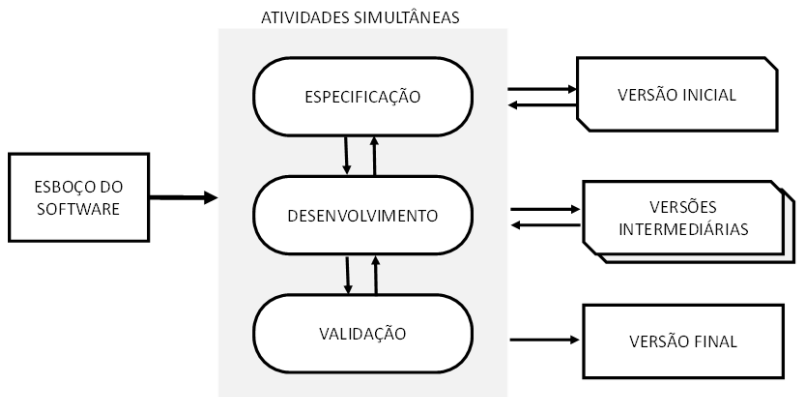
- **Análise e Especificação**: Através de consultas aos usuários do Sistema são estabelecidas as funcionalidades e é realizada a especificação (é realizada a documentação do que foi analisado) do Sistema, com participação e aprovação do cliente.
- **Projeto**: Envolve a abstração do Sistema, alocando recursos para criação do Software; nesta etapa é definida a estrutura de dados, a arquitetura do Software, as *interfaces* gráficas etc.
- **Implementação e teste de unidade**: Na fase de Implementação são realizados os programas (codificação na linguagem de programação escolhida), já o teste Unitário deve ser realizado em cada módulo (ou parte do código) para eliminar falhas de codificação.
- **Integração e Verificação**: Todas as partes (módulos) do Software são integradas e testadas para a entrega ao cliente.
- **Operação e Manutenção**: A partir da efetiva utilização do Software, são realizadas as correções solicitadas pelos usuários e ou implementados novos requisitos que o cliente tenha identificado como necessários.

O modelo em cascata é considerado o modelo mais tradicional e simples, com especificação das atividades de forma clara, além de ser uma base para modelos que surgiram posteriormente e de fácil gerenciamento. Todavia, ao adotar o Modelo em Cascata, o desenvolvimento de um Software pode se estender ao longo de meses, dependendo da complexidade do projeto, uma vez que as tarefas são realizadas de forma sequencial e o atraso em uma das etapas reflete nas demais. Um cliente pode ter que esperar diversos meses pela entrega do Software, o que pode gerar insatisfação com o produto final. Além disso, há apenas uma fase de especificação de requisitos. Diante desse cenário, uma possibilidade é adotar outro modelo de processo de software, o modelo incremental.

O Modelo Incremental é um modelo iterativo que visa, a partir de requisitos iniciais, criar pequenas versões do Software, que vão sendo entregues ao cliente, e posteriormente expandir o Software em novas versões até o sistema ideal ser totalmente construído, segundo Sommerville (2011). Nesse modelo, uma versão é um incremento. Cada incremento (ou versão) incorpora parte da funcionalidade requisitada pelo cliente. No modelo incremental ao invés do cliente receber o Software em uma única entrega, ele receberá

"pedaços" do Software (versões), a cada incremento, até que o Software seja desenvolvido por completo. Esses "pedaços" são módulos que acrescentam, ou melhoram as funcionalidades do sistema. Cada incremento (entrega) é lançado como uma nova versão, do software, até que se atinja a versão final. Na Figura 1.5, observe um esquema do Modelo Incremental. Veja que as atividades: Especificação, Desenvolvimento e Validação são realizadas de forma intercalada e não separada, e, conforme Sommerville (2011), deve ser realizado um rápido *feedback* entre as atividades simultâneas. Em cada versão (incremento) é realizado todo o ciclo de desenvolvimento de Software (do Planejamento aos Testes) e cada versão produzida é um Sistema funcional (que pode entrar em operação), mesmo ainda não estando completo (pode faltar vários requisitos).

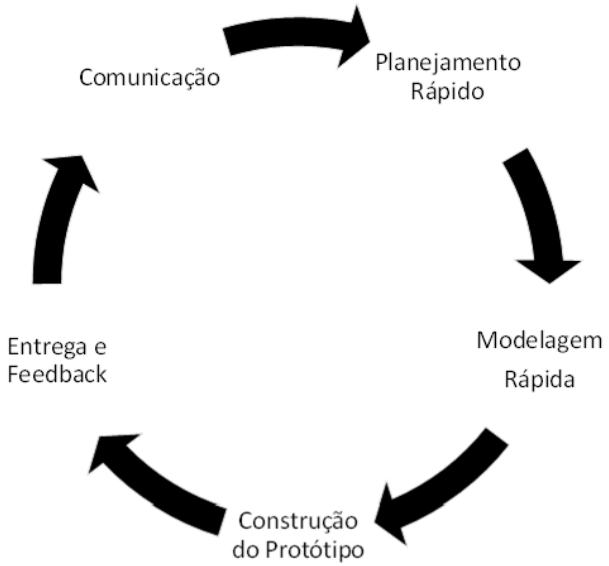
Figura 1.5 | Modelo de Processo Prescritivo: Modelo Incremental



Fonte: adaptada de Sommerville (2011, p. 22).

Outro modelo que pode ser adotado em um projeto é o Modelo de Processo Evolucionário. O Modelo de Processo Evolucionário produz uma versão cada vez mais completa do Software. Pressman (2016) afirma que esse tipo de modelo é iterativo e evolui ao longo do tempo, o que se alinha perfeitamente a um o projeto do Software, pois os requisitos do negócio e do produto não são estáveis, eles mudam (evoluem) e o Software pode ser desenvolvido pensando nesta evolução. No Modelo de Processo Evolucionário aparecem dois Modelos: Prototipação e Espiral. A Figura 1.6 ilustra o esquema do Modelo Prototipação.

Figura 1.6 | Modelo de Processo Prescritivo: Modelo Evolucionário- Prototipação



Fonte: Pressman (2016, p. 45).

Na Figura 1.6 podemos observar as seguintes atividades: Comunicação, Planejamento Rápido, Modelagem Rápida, Construção do Protótipo e Entrega e Feedback. O Modelo de Prototipação começa a partir da Comunicação, identificando quais são os objetivos e as funcionalidades do Software. No Planejamento Rápido são determinados os requisitos que serão Modelados (e quais serão "deixados" para serem implementados mais tarde) e que já podem ser Modelados e Construídos. Após a Entrega, o cliente dá um Feedback e a equipe de desenvolvimento irá aprimorar os requisitos.

O modelo de Prototipação é útil para se apresentar uma versão inicial do software. Com essa versão inicial é possível fazer experimentações com usuários, testar funcionalidades, integração de componentes e sistemas, validar requisitos, dentre outras vantagens. É importante destacar que essa técnica pode ser utilizada em quaisquer partes do Processo de Software, pois a Prototipação ajuda no entendimento do que será construído para o cliente.

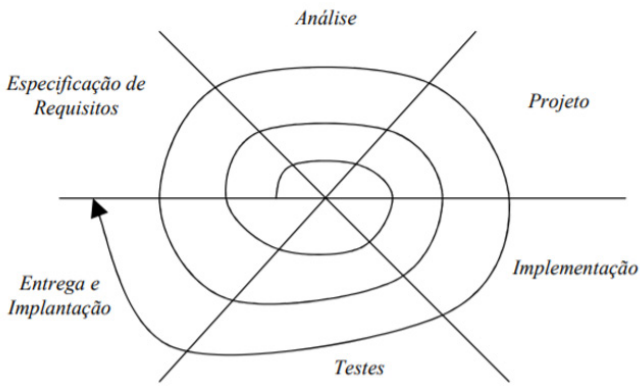
Assimile

O Modelo de Prototipagem tem seu início quando são definidos os requisitos do Software pelo cliente e o Analista de Sistemas. Após isso, é criado um protótipo do Software para que o cliente avalie e realize

testes de funcionalidades de modo a verificar se atende suas necessidades (requisitos), e servindo de orientação aos desenvolvedores. O cliente pode identificar alguma modificação, inclusão ou melhoria de alguma funcionalidade e solicitar os ajustes ao Analista de Sistemas. Ao final, esse protótipo será o produto (Software Final).

O Modelo Espiral (um tipo do Modelo Evolutivo) é iterativo como a prototipação, mas utiliza os aspectos sistemáticos e controlados do Modelo Cascata, conforme afirma Pressman (2016). O objetivo do Modelo Espiral é fornecer um rápido desenvolvimento de versão, que a cada ciclo possa gerar versões mais completas.

Figura 1.7 | Modelo de Processo Prescritivo: Modelo Evolucionário- Espiral



Fonte: Falbo (2012, p. 14).

Pressman (2016) afirma que os Modelos Prescritivos Concorrentes são representados graficamente por uma série de tarefas e técnicas maiores e estados associados a elas e que são utilizados como um paradigma para o desenvolvimento de aplicações Cliente/Servidor. Este Modelo permite que a equipe de desenvolvimento possa representar elementos concorrentes e iterativos de qualquer um dos Modelos de Processos de Software (integrando diferentes tipos de Modelos de Processos de Software). Os Modelos Concorrentes são utilizados em projetos que envolvem diferentes equipes de desenvolvimento e, conforme Pressman (2016), os planos de projeto devem ser considerados documentos vivos e a evolução de cada Processo deve ser avaliada com frequência e revisada levando em consideração as alterações. Os Modelos de Processos Concorrentes podem ser aplicados a diversos tipos de desenvolvimento de Softwares e, diferentemente do Modelo Cascata, ele

não segue uma sequência de atividades, mas estabelece uma rede de atividades que se movimentam de uma atividade para outra.

Reflita

É fato que o surgimento de novas tecnologias mudam o comportamento da sociedade em usar os dispositivos eletrônicos. Será que os Modelos de Processos Prescritivos podem ser utilizados por uma empresa de desenvolvimento de Software? Ainda há espaço para esses Modelos?

Modelo de Processo Especializado

Os Modelos de Processos Especializados utilizam muitas das características de um ou mais Modelos de Processos Prescritivos e são utilizados quando existe a necessidade de uma abordagem mais especializada de Engenharia de Software. Conforme Pressman (2016), os Modelos de Processos Especializados são os seguintes:

- **Modelo Baseado em Componentes:** são utilizados em projetos de Software de Prateleira (Software de Linha), compreende aplicações de componentes de Software previamente empacotados (e vendidos em partes ou completo). São desenvolvidos para poderem ser reutilizados em outros projetos. Um componente é uma parte do independente do Software e pode ser trocado ou alterado.
- **Modelo de Métodos Formais:** compreende um conjunto de atividades que levam à especificação matemática formal do Software, fornecendo mecanismos para a descoberta e a eliminação de muitos problemas como a ambiguidade, incompletude e inconsistência. Servem como base para fazer a averiguação do código de programação com o objetivo de descobrir erros (que passariam despercebidos).
- **Desenvolvimento de Software Orientado a Aspecto:** fornece um processo e abordagem metodológica para definir, especificar, projetar e construir aspectos. O código do Software é separado de acordo com sua importância (classes orientadas a objetos é um exemplo de aspecto) e os requisitos são modelados transcendendo várias funcionalidades do Sistema.
- **Modelo de Processo Unificado:** (conhecida também como RUP – *Rational Unified Process*) aproveita as características dos Modelos de Processos tradicionais (Prescritivos), mas implementa alguns princípios da metodologia Ágil (abordada mais adiante nesta seção); é considerado um Modelo iterativo e incremental.

- **Modelos de Processos Pessoal e de Equipe:** o cerne do desenvolvimento de Software está diretamente ligado a toda equipe de desenvolvimento. No Modelo de Processo de Software Pessoal é enfatizada a medição pessoal do que foi produzido (do artefato gerado e a qualidade resultante). O Modelo de Processo de Software de Equipe objetiva a criação de uma equipe autodirigida, que se organiza por si mesma com a finalidade de produzir um Software com alto padrão de qualidade.

Modelo de Desenvolvimento Ágil

Com o rápido surgimento de novas tecnologias, o processo de negócios também foi atingido por essa velocidade, o que demanda maior velocidade no desenvolvimento de software. Nesse cenário surge uma nova forma de desenvolver Software através da Metodologia Ágil, que traz um formato mais flexível e dinâmico nos Processos de Softwares.

O Desenvolvimento Ágil procura resolver alguns problemas da Engenharia de Software, oferecendo benefícios importantes. As etapas de Levantamento de Requisitos, Análise e Projeto são muito demoradas e, de acordo com Pressman (2016), o Desenvolvimento Ágil é uma resposta ao rápido desenvolvimento de Software (os clientes querem ver resultados rapidamente) e Aplicativos (onde novas funcionalidades são agregadas, na medida que surgem novas ideias ou necessidades), tornando desta maneira o desenvolvimento mais flexível e atendendo as necessidades do cliente com mais rapidez.

Pressman (2016) afirma que o princípio do Desenvolvimento Ágil é focado nas entregas, priorizando também a comunicação entre os envolvidos de forma ativa e contínua para realizar entregas incrementais (procurando a satisfação do cliente). Sommerville (2011) destaca os seguintes princípios do Desenvolvimento Ágil:

- **Envolvimento do Cliente**: o cliente deve ter um forte envolvimento no processo de desenvolvimento do Software fornecendo os requisitos e avaliando o que foi desenvolvido.
- **Entrega Incremental**: o Software é desenvolvido com incrementos, o cliente indica os novos requisitos que devem ser acrescidos.
- **Pessoas e Não Processos**: a equipe possui liberdade de desenvolvimento, explorando ao máximo a capacidade dos desenvolvedores (não estão presos a Processos Prescritivos).
- **Aceitar as Mudanças**: os requisitos podem ser alterados, o projeto deve ser elaborando pensando nesta possibilidade.

- **Manter a Simplicidade**: a complexidade deve ser banida, a equipe deve trabalhar de forma simples e ativa.

Segundo Pressman (2016), o Processo de Desenvolvimento Ágil visa reduzir drasticamente a documentação, tornando o processo de desenvolvimento flexível e reduzindo a burocracia (presente em outros Modelos de Processos de Softwares). Nesse universo, dois Métodos Ágeis se destacam: **XP** (*Extreme Programming*) e **Scrum**.

No Método (ou metodologia) XP o *Feedback* é constante, a abordagem de desenvolvimento é incremental e a comunicação entre os envolvidos é primordial. Pressman (2016) destaca quatro atividades metodológicas que precisam ser seguidas: o Planejamento, o Projeto, a Codificação e os Testes; o desenvolvimento do Software deve ser padronizado e com o trabalho sendo realizado em pares e o cliente (um representante indicado) deve estar sempre presente para esclarecer dúvidas (ele faz parte da equipe de desenvolvimento).

A metodologia Scrum determina um processo de desenvolvimento iterativo e incremental, e pode ser utilizado em processos gerenciais. Esse método define um conjunto de regras e práticas de gestão, para alcançar o sucesso dos projetos como, por exemplo, o trabalho em equipe e comunicação melhorada. Pressman (2016) destaca que o Scrum possui as seguintes atividades metodológicas: Requisitos, Análise, Projeto, Evolução e Entrega; e em cada atividade ocorrem as seguintes tarefas principais:

- *Backlog*: lista com prioridades dos requisitos (das funcionalidades) do projeto, na qual um item pode ser adicionado ou eliminado a qualquer momento (essas são as alterações) e o gerente do produto deve registrar e atualizar as prioridades.
- *Sprints*: são unidades de trabalho para atingir um requisito (estabelecido no *Backlog*) e precisa ser ajustado dentro do *Time Box* (Janela de Tempo) para definir os prazos de entrega.
- Existe, ainda, uma reunião de planejamento na qual o *Product Owner* (dono do produto) prioriza os itens do *Product Backlog* e a equipe seleciona as atividades que ela será capaz de implementar durante o *Sprint* que se inicia.
- Reuniões *Scrum*: são reuniões breves, geralmente, de 15 minutos, chamadas *Daily Meeting* e realizadas diariamente (geralmente no início da manhã), nesta reunião são realizadas três perguntas chaves (para cada integrante da equipe): (i) O que você realizou desde a última reunião de equipe?; (ii) Quais foram os obstáculos encontrados?; (iii) O que planeja realizar até a próxima reunião?

Quando o projeto inicializa, são definidas as ideias e funcionalidades iniciais do produto a ser desenvolvido, estas ideias são chamadas Histórias e o conjunto de todas as Histórias forma o *Product Backlog*. No Método Scrum, geralmente as reuniões são conduzidas pelo *Scrum Master* (o líder da equipe) que conduz o processo e realiza avaliações das respostas de cada integrante da equipe, detectando de forma precoce eventuais problemas, como atrasos ou dificuldade de entendimento de algum requisito. Conforme Pressman (2016), no término do *Sprint* os requisitos são concluídos e o funcionamento é avaliado, melhorando o processo para a *Sprint* sequencial. Cada Sprint se encerra com um incremento ao produto (ou *Product Backlog*).

Existem ainda outros métodos de Desenvolvimento Ágil, Pressman (2016) lista os seguintes: Método de Desenvolvimento de Sistemas Dinâmicos (DSDM), Modelagem Ágil e Processo Unificado Ágil. Todos os métodos ágeis enfatizam a colaboração humana e a auto-organização como elementos chaves.

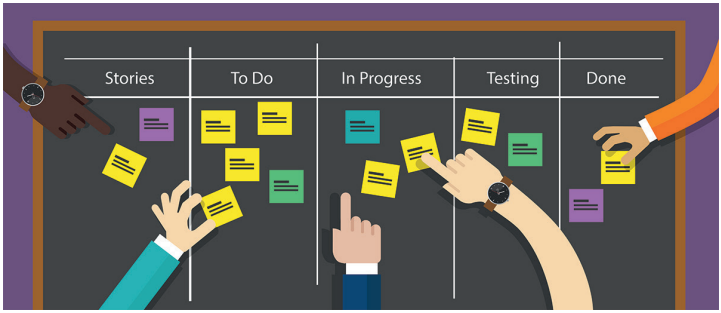
Todas os métodos de desenvolvimento Ágil são baseados no Manifesto Ágil, mas o que é isso? O Manifesto Ágil possui quatro valores fundamentais, exemplificando:

- Primeiro: indivíduos e suas interações são mais importantes que os processos e ferramentas.
- Segundo: software que funciona é mais importante que uma documentação vasta.
- Terceiro: A colaboração do cliente é mais importante que negociação de contratos.
- Quarto: responder às mudanças solicitadas é mais importante que seguir um plano.

Exemplificando

Na metodologia Scrum sempre é montado um quadro (board) para acompanhar as tarefas. Esse quadro pode ser adaptado para a realidade de cada equipe de desenvolvimento. Observe um exemplo de Quadro Scrum, na Figura 1.8.

Figura 1.8 | Quadro Scrum



Fonte: https://www.shutterstock.com/pt/image-vector/scrum-agile-board-260061800.

No quadro são inseridas as atividades a serem resolvidas:

- Stories: é a descrição das necessidades do cliente (sob o ponto de vista deste usuário). É o requisito explicado pelo próprio usuário.
- To Do (A Fazer): são as tarefas que deverão ser realizadas.
- In Progress (em andamento): são todas as tarefas que estão em andamento (mas não estão finalizadas).
- Testing: são os módulos que estão na fase de teste.
- Done (Pronto): é a relação do que já foi finalizado no Sistema.

O quadro do Scrum, com o passar do tempo, fica cheio, refletindo o fluxo de trabalho da equipe de desenvolvimento.

Nesta seção apresentamos os Modelos de Processos de Software e seus diferentes tipos: Modelos Prescritivos, Modelos Especializados e Modelos Ágeis. Todos os Modelos apresentados são amplamente utilizados nas empresas de desenvolvimento de Software. O que determinará qual o tipo de modelo a ser usado são dois fatores: o tipo de Software que deverá ser produzido e a experiência dos Analistas de Sistemas (envolvidos nos projetos).

É importante que você tenha conhecimento sobre esses Modelos de Processos de Software. Em uma entrevista de emprego na área de TI esse assunto é sempre abordado para: determinar o seu grau de conhecimento sobre esses modelos, visto que a grande maioria dos modelos apresentados destacam os fatores: comunicação e trabalho em equipe. Esses dois fatores são expostos como itens indispensáveis no desenvolvimento de um Software.

Vamos continuar buscando conhecimentos no mundo da Análise e Modelagem de Sistemas?

Continue seus estudos, oportunidades boas estão esperando por você!

Sem medo de errar

Você é um Analista de Sistemas e recebeu a missão de determinar qual Modelo de Processo de Software será mais indicado para o mais novo projeto de Software. A Software House, onde você trabalha, possui um novo cliente que deseja um Software para sua loja de brinquedos *online*. O Software deverá permitir que os clientes da loja comprem os produtos disponíveis no *site*. Além do *site*, o cliente deseja um aplicativo, caso o usuário queira acessar a loja para comprar ou alugar brinquedos (o aluguel somente estará disponível na versão em aplicativo).

Para responder qual Modelo de Processo de Software será mais indicado para o mais novo projeto de Software, antes devemos observar dois detalhes sobre o Software a ser desenvolvido: primeiro, o cliente deseja um site (um *e-commerce*) e, segundo, um aplicativo.

Precisaremos de uma equipe qualificada para realizar o desenvolvimento. A equipe deverá ser subdivida em duas partes: uma para o site e outra para o aplicativo. Há um porém: não foi mencionada qual plataforma deverá ser desenvolvida o aplicativo. Caso considerarmos os Sistemas Operacionais Android e iOS, deveremos ter uma equipe para cada Sistema Operacional.

Antes de propor um modelo de processo de software é preciso compreender a complexidade do problema e alinhar os prazos e valores com o cliente. Algumas perguntas que devem ser feitas antes da escolha, são:

1. O cliente tem pressa?
2. A equipe de desenvolvimento é grande o suficiente para trabalhar neste projeto?
3. A equipe domina toda a tecnologia envolvida para o desenvolvimento do site e do aplicativo?

Esse projeto é gigante, visto que o cliente deseja um *site* e um aplicativo; nesse caso, existe uma grande probabilidade de um modelo baseado nos Processos Ágeis ser o mais recomendável, entretanto, a participação do cliente é essencial para o sucesso desta metodologia. Um dos princípios dos Processos Ágeis é dividir o Software para entregá-lo em partes menores. O cliente não precisa esperar o site e os aplicativos ficarem totalmente prontos para ver o resultado final, mas pode participar ativamente de todo o processo de desenvolvimento.

Uma possível abordagem para esse projeto é adotar a metodologia Scrum. Nessa metodologia, entregas devem ser feitas a cada *Sprint*; dessa forma o cliente saberá o que vai receber e quando receberá. Os requisitos mais importantes podem ser entregues primeiro possibilitando que tanto o *site* quanto o aplicativo comecem a operar. Outro ponto importante é a

construção do quadro que ajudará a organizar o cronograma e as equipes de trabalho. Veja no Quadro 1.2 um possível quadro que pode ser usado no projeto. Esse quadro apresenta algumas tarefas pertinentes ao desenvolvimento do *site*. Como podemos observar no *Backlog,* estão previstas as tarefas Gerar *Interfaces* do *Site* (conhecidas como *Wireframes*), Definir a Paleta de Cores do Site, Definir a Estrutura do Banco de Dados.

Quadro 1.2 | Exemplo de quadro Scrum para o projeto

ATIVIDADES DO BACKLOG			
ITENS DO BACKLOG	A FAZER	EM ANDAMENTO	PRONTO
Gerar Interfaces do Site (Wireframes)			
Definir Paleta de Cores			
Definir Estrutura do Banco de Dados			
.....			

Fonte: elaborado pela autora (2020).

Agora é com você! Tente propor outro modelo de desenvolvimento para o projeto. Faça uma lista de vantagens e desvantagens entre esse processo e a Metodologia Ágil. Pressman (2016) afirma que o melhor Processo de Software é aquele próximo à equipe de desenvolvimento, ou seja, um cenário ideal é propor um Processo de Software que se adapte às necessidades da equipe.

Boa jornada de estudos!

Faça valer a pena

1. Compreende um conjunto de atividades que levam à especificação matemática formal do Software, fornecendo mecanismos para a descoberta e a eliminação de muitos problemas, como a ambiguidade, incompletude e inconsistência. Servem como base para fazer a averiguação do código de programação com o objetivo de descobrir erros (que passariam despercebidos).

Assinale a alternativa correta referente ao nome deste Modelo de Processos.

a. Modelo Prescritivo.

b. Modelo de Processo Unificado.

c. Modelo Scrum.

d. Modelo Cascata.

e. Modelo de Métodos Formais.

2. Pressman (2016) afirma que o Modelo Evolucionário é interativo e evolui ao passar o tempo. O projeto do Software evolui, pois os requisitos do negócio e do produto não são estáveis, eles mudam (evoluem) e o Software pode ser desenvolvido pensando nesta evolução.

Assinale a alternativa que demonstra dois tipos de Modelos Evolucionários.

a. Scrum e XP.

b. Cascata e Incremental.

c. Espiral e Protótipo.

d. RUP e Métodos Formais.

e. Baseados em Componentes e Baseados em Objetos.

3. Os modelos de Processos Prescritivos possuem a finalidade de estruturar e ordenar o desenvolvimento de um Software e, conforme Pressman (2016), as tarefas ocorrem de forma sequencial, com diretrizes bem definidas, pois indicam uma série de atividades metodológicas, ações, tarefas, artefatos, garantias de qualidade e mecanismos de controle de mudanças para cada projeto.

Sobre os Processos Prescritivos, analise as afirmativas a seguir.

I. Os Modelos Concorrentes são utilizados em projetos que envolvem diferentes equipes de desenvolvimento.

II. O Modelo de Prototipagem tem seu início quando são definidos os requisitos do Software pelo cliente e o Analista de Sistemas.

III. O Modelo de Processo Evolucionário produz uma versão cada vez mais completa do Software.

IV. O Modelo Espiral é interativo como a prototipação, mas utiliza os aspectos sistemáticos e controlados do Modelo Incremental.

Analisando as afirmativas apresentadas, é correto o que se afirma em:

a. II e III, apenas.

b. I e IV, apenas.

c. I e II, apenas.

d. I, II e III, apenas.

e. I, II, III e IV.

Referências

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados.** 6. ed. [*S. l.*]: Pearson Addison Wesley, 2011.

ENGHOLM JR., H. **Engenharia de Software na Prática.** São Paulo: Novatec, 2010.

FALBO, R. A. **Engenharia de Requisitos:** Notas de Aula. Universidade Federal do Espírito Santo, Vitória, 2012.

FONSECA FILHO, C. **História da computação:** o caminho do pensamento e da tecnologia. Porto Alegre: EDIPUCRS, 2007. Disponível em: http://www.pucrs.br/edipucrs/online/historia-dacomputacao.pdf. Acesso em: 31 jan. 2020.

PAULA FILHO, W. P. **Engenharia de software:** produtos. 4. ed. Rio de Janeiro: LTC, 2019.

PFLEEGER, S. L. **Engenharia de software:** teoria e prática. Tradução de Dino Franklin. Revisão técnica Ana Regina Cavalcanti da Rocha. 2. ed. São Paulo: Prentice Hall, 2004.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software**: uma abordagem profissional. Tradução de João Eduardo Nóbrega Tortello. Revisão técnica: Reginaldo Arakaki, Julio Arakaki e Renato Manzan de Andrade. 8. ed. Porto Alegre: AMGH, 2016.

PRESSMAN, R. S. **Software engineering:** a practitioner's approach. 6. ed. Nova York: McGraw-Hill, 2005.

ROTH, R. M.; DENNIS, A.; WIXOM, B. H. **Análise e projeto de sistemas.** 5. ed. Rio de Janeiro: LTC, 2014

SOMMERVILLE, I. **Engenharia de software.** 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TANENBAUM, A. S. **Sistemas operacionais modernos.** 2. ed. [*S.l.*]: Prentice Hall, 2003.