ENGENHARIA DE *SOFTWARE*

Aline Zanin

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Conceitos da Engenharia de Software

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Reconhecer o histórico e os conceitos fundamentais da Engenharia de Software.
- Analisar a evolução do desenvolvimento de software.
- Identificar a importância da Engenharia de Software.

Introdução

Por muitos anos a Engenharia de Software foi utilizada com o objetivo de criar software de qualidade dentro dos custos e dos prazos estimados pelo cliente, evitando desperdícios de tempo, esforços, direções erradas e atrasos. A criação de software foi subestimada e realizada sem nenhuma metodologia, gerando erros em sistemas, como problemas de cálculos e perdas financeiras e de tempo. Nesse período, podemos dizer que houve a crise do software. Com isso, em 1967, a Organização do Tratado do Atlântico Norte (OTAN) designou o termo Engenharia de Software para adequar o processo de desenvolvimento de software com metodologias já utilizadas em outras Engenharias. Uma série de metodologias e técnicas passaram a ser utilizadas antes, durante e depois da criação de software. Dados históricos apontam que houve uma diminuição brutal nos problemas em software após a adoção dessas metodologias, fazendo com que a indústria de software pudesse entregar sistemas com maior qualidade, em menos tempo e com custos reduzidos de manutenção.

Neste capítulo, você irá adquirir conhecimentos fundamentais para avançar no aprendizado sobre Engenharia de Software. Iremos abordar inicialmente conceitos básicos sobre o que é essa Engenharia, sua história e a importância na indústria.

Histórico e conceitos fundamentais da Engenharia de Software

A Engenharia de Software é uma disciplina da Engenharia, mais especificamente da Ciência da Computação, que estuda todos os processos envolvidos no desenvolvimento de software, uma atividade complexa que envolve a realização conjunta de diversas atividades distintas, as quais exigem habilidades multidisciplinares e, por consequência, trabalho colaborativo de um grande grupo de profissionais (SOMMERVILLE, 2011).

A Engenharia de Software é uma área de grande importância, uma vez que as pessoas e a sociedade como um todo estão a cada dia mais dependentes de software. Por isso, faz-se necessário que seja produzido software mais confiável e de forma mais rápida a cada dia. Além disso, para as empresas desenvolvedoras de sistemas, geralmente é mais barato, a longo prazo, usar métodos e técnicas da Engenharia de Software para o desenvolvimento de sistemas do que desenvolver os sistemas sem documentação e estruturação, uma vez que, desta forma, é desestruturada e dificultada a manutenção do software (SOMMERVILLE, 2011).

Contudo, esta disciplina nem sempre foi alvo de atenção dos profissionais de Tecnologia da Informação. Por muito tempo, o desenvolvimento de sistemas foi realizado sem atenção a processos, metodologias e estruturas organizacionais no que diz respeito a tarefas, atividades e responsabilidades. Essa falta de controle sobre os processos fez com que, muitas vezes, o software fosse entregue aos clientes sem a devida qualidade e com grande número de erros, como problemas de cálculos e perdas financeiras e de tempo. Nesse período, podemos dizer que houve a crise do software. Com isso, em 1967, a OTAN designou o termo Engenharia de Software para adequar o processo de desenvolvimento de software com metodologias já utilizadas em outras engenharias.

A partir desse momento, os profissionais e as empresas de Tecnologia da Informação passaram a preocupar-se mais com os diversos setores que envolvem o desenvolvimento de sistemas, como análise de requisitos, análise de sistemas, desenvolvimento, testes e implantação. Neste contexto, derivaram diversas metodologias, métodos e processos para auxiliar e guiar o trabalho de cada um desses segmentos.

Evolução do desenvolvimento de software

O desenvolvimento de software, bem como outras Ciências, empregou diversas mudanças e adaptações para melhorar, facilitar e adaptar-se ao cotidiano dos profissionais que realizam esse trabalho. As principais evoluções no desenvolvimento de software podem ser classificadas em dois grandes grupos: mudanças processuais e mudanças tecnológicas.

Mudanças tecnológicas

Embora atualmente, quando se fala em software, sejamos remetidos a lembrar de computadores modernos, *smartphones*, *tablets*, etc., o desenvolvimento de software começou muito antes desses dispositivos serem criados, sendo programado por volta de 1725, em cartões perfurados. Posteriormente, surgiram as primeiras linguagens de programação, tais quais as que existem hoje, sendo elas FORTRAN (1955), *List Processor* (LISP) e *Common Business Oriented Language* (COBOL). Posteriormente, surgiram linguagens de programação de alto nível, isto é, que se aproximam mais da linguagem humana, são exemplos: *Java*, *JavaScript*, *Visual Basic*, *Object Pascal* e PHP (PACIEVITCH, 2017).

Junto com as linguagens de programação, foram sendo criados paradigmas para o desenvolvimento de sistemas. Um paradigma nada mais é do que a forma como um sistema é construído e seu desenvolvimento é organizado. Os paradigmas mais conhecidos são o paradigma estruturado e o paradigma orientado a objetos, sendo o paradigma orientado a objetos o mais utilizado atualmente. A programação orientada a objetos é um paradigma em que o software é construído considerando que tudo o que é inserido no programa é um objeto e que esse objeto pertence a uma classe e tem características (atributos) específicas sobre as quais podem ser feitas ações (métodos). Por outro lado, um princípio básico da programação estruturada é que um programa pode ser dividido em três partes que se interligam, sendo elas sequência, seleção e iteração (ABÍLIO, 2017). Na sequência, o código do programa é criado para ser executado de forma sequencial, seguindo estritamente a ordem na qual foi programado. Na seleção, o programa encontra locais onde pode seguir um ou mais caminhos distintos. Na interação, é permitido ao programa executar diversas vezes o mesmo trecho de código.

Fique atento

- Ao programar uma calculadora, quando cria-se um programa e o único fluxo que este pode seguir é receber dois números, somar esses números e mostrar o resultado, faz-se um programa utilizando apenas sequência.
- Quando se insere neste programa a opção de selecionar se deve somar ou subtrair os números, se está programando uma seleção.
- Quando, ao final do cálculo, pergunta-se para o usuário se deseja fazer novamente, se está programando uma interação.

Mudanças de processo

No desenvolvimento de sistemas, além das mudanças tecnológicas, foram ocorrendo mudanças na forma como as empresas se organizam e estruturam o trabalho. A forma tradicional de desenvolvimento de sistemas foi a primeira a ser criada, empregando o ciclo de vida em estrutura de **cascata** (1970), na qual as etapas são executadas de forma sequencial, sem que seja possível retornar de uma etapa posterior para uma etapa anterior.

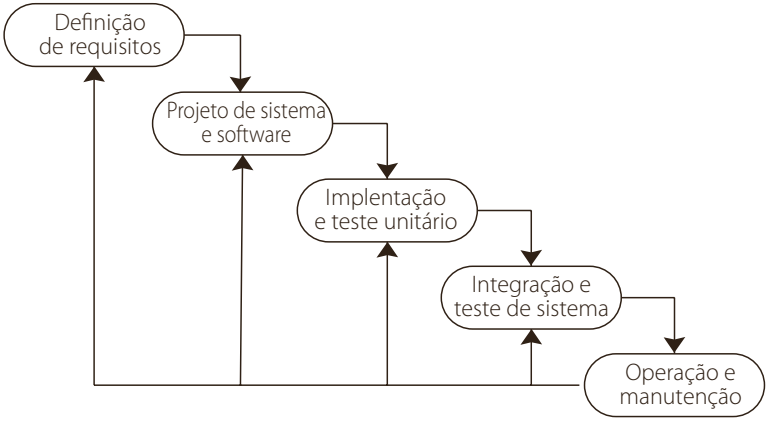

Figura 1. Modelo Cascata

Fonte: Sommerville (2011, p. 20).

Posteriormente, falou-se em desenvolvimento **iterativo e incremental**. Nesse modelo, implementa-se pequenas partes entregáveis do software para que o cliente tenha um *feedback* mais rápido sobre o produto que está sendo desenvolvimento.

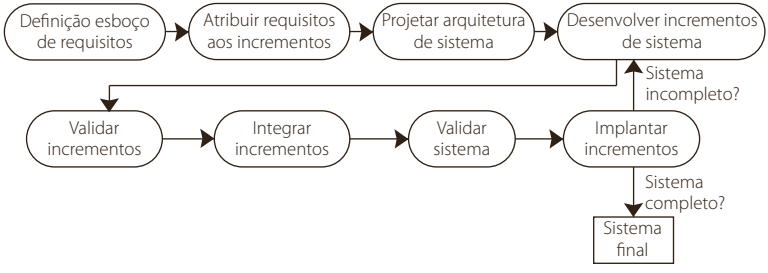

Figura 2. Entrega Incremental.

Fonte: Sommerville (2011, p. 31).

O modelo em **espiral** se assemelha muito ao modelo iterativo e incremental, uma vez que ele também considera pequenas entregas de software e a execução de todas as etapas espiralmente (várias vezes). Contudo, o ciclo de vida espiral considera a presença explícita da análise de riscos como uma das etapas de cada iteração. Nesse processo em espiral, o ciclo de vida do software é representado como uma espiral em que a volta na espiral representa uma fase do processo de software, sendo que a volta mais interna pode preocupar-se com a viabilidade do sistema, o ciclo seguinte, com definição de requisitos, o seguinte, com o projeto do sistema, e assim por diante.

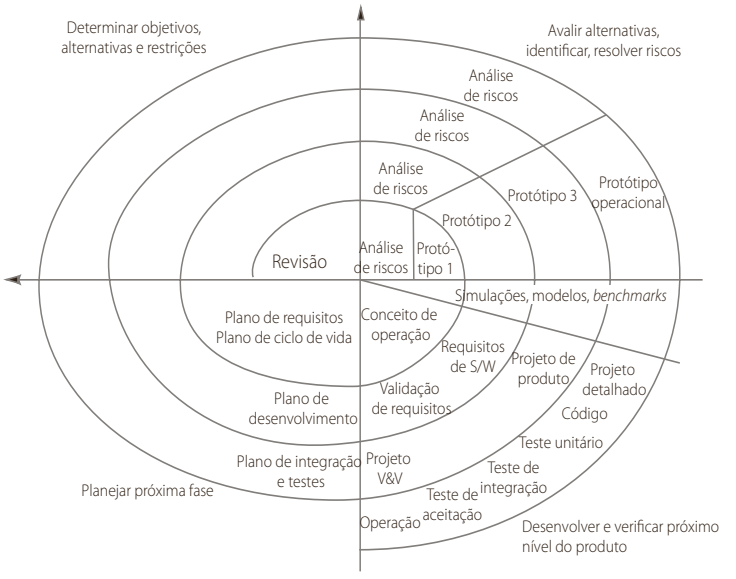

Figura 3. Modelo Espiral.

Fonte: Sommerville (2011, p. 33).

No ano de 2001, um grupo de profissionais de tecnologia da informação lançou um documento chamado Manifesto Ágil para o Desenvolvimento de Sistemas. Deste então popularizaram-se os métodos ágeis de desenvolvimento de sistemas, sendo os mais conhecidos o método Scrun e o método XP. Todos eles têm em comum a aplicação dos valores propostos pelo Manifesto Ágil para o Desenvolvimento de Sistemas, sendo eles: indivíduos e interações são mais que processos e ferramentas, software em funcionamento é mais que documentação abrangente, colaboração com o cliente é mais que negociação de contratos e respostas a mudanças são mais que seguir um plano (BECK et al., 2001).

Todos esses ciclos de vida, somados aos Métodos Ágeis de Desenvolvimento de Sistemas, apresentaram estrutura e organização maiores para o processo de desenvolvimento de sistemas, propiciando melhoria da comunicação entre os envolvidos no processo, seja entre os próprios profissionais de Tecnologia da Informação ou destes com o cliente. Com a adoção de processos e a atenção às evoluções tecnológicas, buscando sempre acompanhar aquilo que o mercado

tem de melhor para oferecer, pode-se atingir maior excelência nos produtos entregues e atender melhor às necessidades do cliente.

Importância da Engenharia de Software

Conforme supracitado, a Engenharia de Software é uma disciplina de Engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até a sua manutenção, quando o sistema já está sendo usado (SOMMERVILLE, 2011). Neste contexto, é clara a importância fundamental da Engenharia de Software, uma vez que, se o processo de desenvolvimento de sistemas envolve diversas atividades distintas e a Engenharia de Software é a disciplina que preocupa-se em estudar e monitorar o bom andamento de todas essas atividades e a integração entre elas, é trivial notar que é baseado na Engenharia de Software o sucesso de um projeto no que tange a sua organização.

Engenharia de Software envolve **planejamento**. Planejamento diz respeito também a pessoas e cronograma de trabalho. Pessoas porque divide as responsabilidades, de forma individual ou coletiva. Cronograma porque conforme o planejamento é que os gestores têm a possibilidade de mensurar o tempo necessário para o desenvolvimento de cada projeto. Engenharia de Software envolve também a preocupação com a **qualidade** do produto. Qualidade, neste contexto, não se refere apenas à entrega de produtos em funcionamento, mas também ao atendimento das necessidades do cliente, e, por isso, a área da Engenharia de Software, que trata de engenharia de **requisitos** ou análise de requisitos, tem importância fundamental, na medida em que é por meio dela que as equipes de desenvolvimento recebem a expectativa do cliente sobre o produto que está sendo desenvolvido para buscar atendê-la.

Além disso, na fase de **desenvolvimento** da programação em si, a Engenharia de Software se faz presente, uma vez que a escolha do processo ideal irá influenciar diretamente no trabalho cotidiano de todos os envolvidos, incluindo a programação. Esse fator ganha relevância ainda maior em times que trabalham em horários distintos ou locais geograficamente distribuídos.

Uma vez entregue o software para o cliente, a Engenharia de Software tem sua importância revelada no momento de realizar a manutenção nesse software. Isto porque, se o sistema tiver sido corretamente planejado, o código do sistema tende a estar mais limpo e com menos defeitos. Isso irá causar menos manutenção e facilitar as manutenções que precisam ser realizadas.

Referências

ABÍLIO, I. *Programação orientada a objetos versus programação estruturada.* Disponível em: <http://www.devmedia.com.br/programacao-orientada-a-objetos-versus-programacao-estruturada/32813>. Acesso em: 17 ago. 2017.

BECK, K. et al. *Manifesto for agile software development.* c2001. Disponível em:<http://agilemanifesto.org/>. Acesso em: 17 ago. 2017.

PACIEVITCH, Y. *História da programação.* Disponível em: <http://www.infoescola.com/informatica/historia-da-programacao>. Acesso em: 17 ago. 2017.

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

Leituras recomendadas

ENDEAVOR BRASIL. *PDCA*: a prática levando sua gestão à perfeição. 2015. Disponível em: <https://endeavor.org.br/pdca/>. Acesso em: 4 nov. 2016.

PEQUENO, C. N.; CARVALHO, M. G. F.; FONTES, V. M. *Redução do consumo de produto químico utilizado em uma linha de produção de uma indústria de pneus*. 2015. Trabalho de Conclusão de Curso (Graduação em Engenharia de Produção)–Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2015.

REY, B. *Como fazer um brainstorming eficiente.* 2013. Disponível em: <http://exame.abril.com.br/carreira/como-fazer-um-brainstorming-eficiente/>. Acesso em: 4 nov. 2016.

RODRIGUES, S. *Crise:* perigo, oportunidade e… papo furado. 2013. Disponível em: <http://veja.abril.com.br/blog/sobre-palavras/lendo-a-lenda/crise-perigo-oportunidade-e-papo-furado/>. Acesso em: 4 nov. 2016.

SILVA, M. D. L. et al. Gestão da produção: estudo sobre a gestão da manutenção na geração de energia e vapor utilizando caldeiras de uma indústria. In: ENCONTRO PARAENSE DE ENGENHARIA DE PRODUÇÃO, 7., 2016, Belém. *Anais*... Belém: [s.n.], 2016.

SLACK, N. et al. *Gerenciamento de operações e de processos*: princípios e práticas de impacto estratégico. 2. ed. Porto Alegre: Bookman, 2013.

SOUZA, T. de J. F. et al. Proposta de melhoria do processo de uma fábrica de polpas por meio da metodologia de análise e solução de problemas. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 35., 2015, Fortaleza. *Anais*... Fortaleza: ABEPRO, 2015. Disponível em: <http://www.abepro.org.br/biblioteca/TN_STP_207_228_27341.pdf>. Acesso em: 4 nov. 2016.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS