DESENVOLVIMENTO ORIENTADO A REÚSO DE *SOFTWARE*

SAGAH SOLUÇÕES EDUCACIONAIS INTEGRADAS

Linha de produção de *software*

Paulo Sérgio Pádua de Lacerda

OBJETIVOS DE APRENDIZAGEM

- Identificar os tipos de especialização de uma linha de produção de *software*.
- Descrever o processo de extensão de uma linha de produção de *software* para criar uma aplicação nova.
- Diferenciar a configuração em tempo de projeto da configuração em tempo de implantação de uma linha de produção de *software*.

Introdução

Na era contemporânea, o *software* é a máquina propulsora dos negócios. Instituições públicas e privadas, além do usuário comum, têm hoje no *software* o meio para fidelização do clientes, oferecimento de serviços, captura de informações para planejamento estratégico, entretenimento, entre outros. Um *software* é um produto abstrato, ou seja, não há limites para seu uso e crescimento.

O cenário atual, principalmente dos negócios, é dinâmico, ágil e sedento por invocações. Esse processo resulta em *softwares* voltados a atender novas necessidades, mas sem exigir o tempo de desenvolvimento do passado. Diante disso, o reúso de *software* é hoje uma tendência nas linhas de produção.

Desenvolver uma linha de produção de *software* permite que as empresas diminuam o tempo de desenvolvimento, barateiem custos, aumentem a eficácia dos processos e atendam um conjunto maior de clientes. Portanto, uma linha de produção de *software* consiste em um sistema de referência-base formado por componentes reutilizados para versões mais específicas.

Neste capítulo, você conhecerá os tipos de especialização de uma linha de produção de *software*, entenderá os requisitos para desenvolver um novo *software* e aprenderá a distinguir o tempo de projeto de configuração do tempo de implantação de uma linha de produto de *software*.

Tipos de especialização

Você já imaginou o tanto de *software* uma empresa fabricante de produtos para área de computação deve produzir? Em fabricantes de impressoras, por exemplo, quantos variados modelos são desenvolvidos para o mercado, cada tipo proporcionando diferentes funções associadas a uma interface de comunicação, como o sistema operacional em uso pelo usuário.

Fique atento

Nesse caso, o termo interface não faz referência à interface do usuário, às telas da aplicação, e sim ao programa usado para estabelecer a comunicação entre a impressora e o computador, também conhecido pelo termo *driver* (LECESSI, 2019).

Uma linha de produção de *software* é definida como um conjunto de funcionalidades comuns e compartilhadas, com um sistema-base (*core*) contento os principais módulos ou componentes genéricos e módulos que podem ser específicos a um determinado segmento, com base nos requisitos do cliente (SOMMERVILLE, 2018).

Nesse caso, o sistema-base contendo as funcionalidades comuns para um determinado segmento é projetado de tal forma que possa ter uma nova configuração e uma adaptação às diferentes necessidades impostas por clientes e dispositivos (SOMMERVILLE, 2018). Muitas vezes, as novas necessidades, geradas principalmente pelo dinamismo dos negócios da era contemporânea, acarretam novos requisitos, e esses novos requisitos resultam em novas configurações de componentes, adição de novos componentes e também modificações de componentes já existentes com o objetivo de atender a esses novos requisitos.

Benefícios da linha de produção de *software*

A escolha pelo desenvolvimento de *software* a partir de uma linha de produção traz diversos benefícios no processo de elaboração de uma aplicação, seja para negócios ou dispositivos. A seguir, elencamos alguns benefícios proporcionados pela linha de produção de *software* (PRESSMAN; MAXIM, 2016).

- Tempo: ao ser desenvolvido, um *software* passa por diversas fases, e seu tempo total de produção costuma ser medido em horas. Um *software* de *game*, por exemplo, leva em torno de 100 a 150 horas para ser desenvolvido, enquanto um aplicativo como o da Uber leva entre 300 a 350 horas. O tempo é determinado pela complexidade do sistema a ser desenvolvido. Porém, quando se usa uma linha de produção, que já conta com um sistema-base, há uma redução no tempo de desenvolvimento do sistema.
- Custo: tende a haver uma redução do custo de projeto quando do reúso de sistemas já existentes. Isso advém do menor tempo necessário para desenvolvimento e da diminuição da complexidade de elaboração de um novo sistema.
- Melhoria do processo de produção: o trabalho a partir de um sistema-base (*core*) ajuda no desenvolvimento de outros sistemas mais específicos, pois grande parte do sistema já existe, não havendo a necessidade de desenvolvê-lo do zero. Esse processo proporciona, por exemplo, redução na fase de teste, pois os testes já realizados na parte existente também podem ser reusados, resultando na redução do tempo total de desenvolvimento da aplicação.
- Aprendizado: por conhecer muito bem grande parte do sistema, os engenheiros de *software* têm mais facilidades para desenvolver novos módulos, criar inovações e produzir mais rapidamente novos sistemas.

Aplicação-base

Uma aplicação-base é desenvolvida para simplificar o reúso no desenvolvimento de novas aplicações mais específicas. Geralmente, projeta-se um linha genérica com base na identificação de requisitos comuns de um determinado segmento (PRESSMAN; MAXIM, 2016). Vamos exemplificar com base em um sistema escolar.

Embora a regra de negócio deva ser diferente entre as diversas escolas, muitos módulos do sistema são comuns entre elas, como um módulo de administração da vida acadêmica do aluno, módulo de registro de colaboradores, módulo de secretaria, etc. Nesse contexto, se você conta com um sistema-base capaz de receber novas configurações, adições de novos módulos ou alterações de módulos existentes, o processo de citação de diversos sistemas escolares para distintas escolas torna-se mais fácil.

Uma aplicação-base é composta pelos seguintes itens (SOMMERVILLE, 2018).

- Núcleo: o núcleo do sistema é formado pelos principais componentes comuns a um determinado segmento. Trata-se do pilar da aplicação, e geralmente sofre bem pouca alteração quando uma nova linha de *software* é produzida.
- Componentes mutáveis: são os componentes alterados com novas configurações, a fim de atender aos requisitos da nova aplicação.
- Componentes específicos: são aqueles configurados ou criados para atender a um domínio em particular.

A Figura 1 ilustra a estrutura hierárquica de um sistema-base. Note que os módulos mais específicos estão na camada mais superior do sistema, enquanto os internos representam seus pilares.

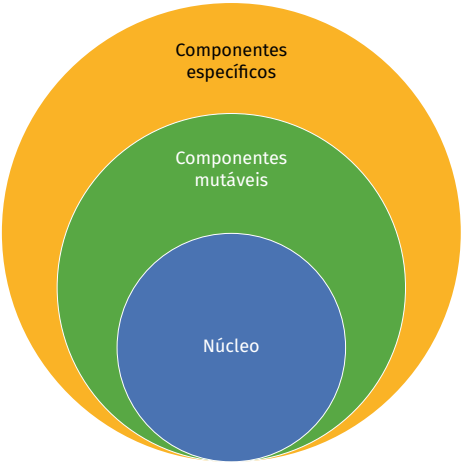

Figura 1. Estrutura de um sistema-base.
Fonte: Adaptada de Sommerville (2018).

Tipos de especialização

Uma linha de produção de *software* está associada a alguns tipos de especialização, discriminados a seguir (SOMMERVILLE, 2018).

- Plataforma: nos tempos da internet, as aplicações são desenvolvidas de acordo com cada plataforma. Hoje, as mais conhecidas são Mac OS, Windows e Linux. O Swift, por exemplo, é um linguagem para desenvolver para aplicações de plataforma Mac OS como iOS, Apple TV e Apple Watch. Já as linguagens Java e Kotlin são usadas para criar aplicações para a plataforma Android de forma nativa. Há novos modelos de desenvolvimento que se baseiam em uma única linguagem, como Javascript, e permitem o desenvolvimento para as plataformas Android e iOS, como React.

Saiba mais

O React é uma biblioteca de desenvolvimento baseada na linguagem Javascript que permite a criação de aplicações tanto para Android quanto para iOS, mas usando uma única linguagem. Na página oficial do React na internet, na Aba "Documentação", você encontra um artigo de introdução e visão geral de seus recursos relacionados (REACT, 2021).

- Ambiente: pode influenciar as modificações de uma aplicação, já que um sistema pode existir em diferentes versões, dependente do *hardware* usado. Como exemplo, Sommerville (2018) menciona que um sistema de emergência da polícia por rádio pode ter a necessidade de incorporar um sistema de criptografia para implementação de segurança. Os componentes de linha de produção de *software* baseados em ambiente devem refletir as características e as funcionalidades associadas ao equipamento de *hardware*, por exemplo.
- Funcionalidade: baseia-se naquilo que o sistema deve executar, ou seja, nos seus requisitos funcionais. Muitas aplicações podem ser modificadas para atender a diferentes contextos, mesmo dentro de um domínio especificado. Um sistema acadêmico, por exemplo, pode ter módulos modificados e outros incorporados se desenvolvidos para uma universidade pública ou particular. Mesmo entre as particulares, o sistema pode ter módulos funcionais modificados, pois esses módulos

devem atender aos requisitos específicos de cada universidade, embora a base do sistema seja a mesma no desenvolvimento das aplicações.

- Processos: dentro de uma organização, os processos determinam o *modus operandi* de uma empresa. Processos podem ser semelhantes, mas podem ter especificações. O mesmo sistema de *delivery*, por exemplo, pode ter mais opções em uma empresa do que em outra. Suponha que um determinado processo da empresa A de entregas de compras só seja possível via agendamento e retirada fisicamente na própria empresa, enquanto na empresa B isso pode ser feito por correio ou carreto particular. Embora os processo de entrega sejam semelhantes, há pontos específicos associados à rede de negócio tanto da empresa A quanto da empresa B.

Mas como uma linha de produção de *software* é capaz de atender aos requisitos de cada negócio? Detalhes de como analisar as necessidade de criar uma nova aplicação serão examinados no próximo tópico.

Extensão de uma linha de produção de *software*

Uma linha de produção de *software* é caracterizada por trazer alguns benefícios às empresas que adotam essa opção, como reúso de *software*, melhor qualidade no desenvolvimento, diminuição nos custos do projeto, aumento de produtividade e integração facilitada (PRESSMAN; MAXIM, 2016). Além desses, há ainda benefícios intangíveis, que são mais complexos de mensurar, como aceitabilidade por parte dos profissionais, satisfação pessoal, satisfação do cliente, etc.

Dentre os benefícios incorporados no processo na linha de produção, o reúso é o fator mais relevante, pois acelera quase todos os processos. Nesse âmbito, há dois tipos de reúso (KEYES, 2003):

- externo: aplicado para gerar novas aplicações;
- interno: utilizado para gerar novas versões da mesma aplicação.

Entretanto, independentemente do modo de reúso a ser aplicado, ao criar uma nova aplicação diversos processos devem ser considerados, como os citados a seguir (SOMMERVILLE, 2018).

- Elencar requisitos dos *stakeholders*: *stakeholders* são aqueles impactados pelas ações da empresa, como clientes, gestores, imprensa,

colaboradores. Ao invés de iniciar o processo de criação de uma nova aplicação pelo processo convencional, ou seja, pela análise de negócio, requisitos, etc., utiliza-se a análise do sistema já pronto. Nesse caso, o sistema é apresentado aos *stakeholders* e os requisitos de necessidade elencados por eles são analisados.

- Selecionar o sistema existente mais adequado aos requisitos: geralmente, a versão existente mais próxima dos requisitos identificados é o sistema-base de partida para as novas versões. Essa ação reduz consideravelmente o tempo de desenvolvimento.
- Renegociar os requisitos: a validação de requisitos é um processo longo e às vezes enfadonho, pois muitos deles são identificados na avaliação do usuário e podem variar conforme o uso. Contudo, um projeto não é estático, e ao longo da execução novos requisitos podem ser identificados e então renegociados com o cliente, já que a implementação de novos requisitos pode dilatar o prazo de desenvolvimento e exigir modificações no sistema-base, por exemplo.
- Adaptar o sistema existente: para satisfazer as necessidades da nova aplicação, muitas vezes é preciso adaptar módulos já existentes, e vez de desenvolver um módulo totalmente do zero. Essas modificações ou adaptações são realizadas para adequação dos novos requisitos.
- Entregar um novo membro da família de produtos: mesmo após a entrega do produto final ao cliente, algumas modificações podem ser necessárias. Elas refletem as exigências dos ambientes onde serão implantadas. Nesse caso, como todo sistema de *software* é evolutivo, deve-se preservar as características principais para aplicação em novos módulos ou versões.

A Figura 2 ilustra o fluxo de processos na criação de uma nova aplicação usando uma linha de produção de *software*.

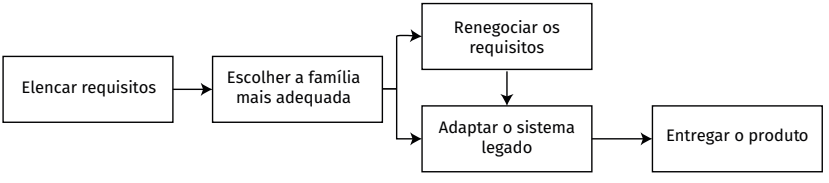

Figura 2. Fluxo de processos de novo produto.
Fonte: Adaptada de Sommerville (2018).

Segundo essa esquematização, nota-se que, na criação de uma nova linha de produto, quanto mais a versão já existente se aproximar dos requisitos identificados pelos *stakeholders*, mais rápido será o processo de desenvolvimento, pois menos trabalhos será realizado. Os requisitos são essenciais para a satisfação final dos *stakeholders*, e quanto mais detalhados, mais resultam em um produto que atenda fielmente às suas expectativas.

Todavia, um linha de produção de *software* é dinâmica, ou seja, não é estática, e pode sofrer modificações diversas, como exclusão de componentes, adaptação de componentes, incorporação de novos componentes, novas restrições de regra de negócio incorporadas aos componentes, etc. Esses detalhes são discutidos no próximo tópico.

Configuração em tempo de projeto e em tempo de implantação

As modificações em um processo de linha de produção de *software* podem ser realizadas em diferentes tempos: em tempo de projeto e em tempo de implantação (SOMMERVILLE, 2018). Em tempo de projeto, as modificações são realizadas pela empresa no sistema-base de uma linha de produtos para geração de uma nova linha. Nesse caso, a empresa incorpora, exclui e modifica componentes para criar o novo produto a ser entregue ao cliente. Como exemplo, podemos citar a linha de produtos Windows. Ao longo de tempo, a empresa Microsoft foi evoluindo o produto para adequá-lo aos novos requisitos elencados pelo uso, como a integração com a internet e aos serviços associados a dispositivos como *smartphone*, além dos produtos associados a clientes corporativos, como a linha Server.

Nas modificações em tempo de implantação, por sua vez, o mais adequado é usar um sistema genérico que possa ser modificado para atender às necessidades do cliente. Nesse caso, requisitos como novas regras de negócio e o ambiente operacional são incorporados ao sistema. Sistema como os de gestão de relacionamento com o cliente (*customer relationship management* — CRM), por exemplo, usados para elencar estratégias de relações de atendimento, costumam ser modificados em tempo de implantação, pois muitas vezes precisam passar por alterações para satisfazer requisitos identificados diretamente junto ao cliente.

De acordo com Sommerville (2018) e Pressman e Maxim (2016), são os seguintes os níveis que determinam os estágios de modificações de linha de produto de *software* em tempo de implantação.

- Escolha de componentes: muitas vezes os sistema são desenvolvidos em outros tempos, e por uma questão de adequação precisam ser modificados para armazenamento de novos requisitos. Hoje, muitos sistemas precisam armazenar informações relacionadas a exames de máquina, por exemplo, como tomógrafos. Nesse caso, somente um componente é inserido no sistema para gerenciar as imagens, armazená-las, lê-las e integrá-las ao prontuário do paciente.
- Definição de fluxo de trabalho e de regras: nessa parte, identificam-se as regras que validam as informações do sistema e o fluxo de processamento dessas informações.
- Definição de parâmetros: os parâmetros são dados utilizados no sistema para determinar, por exemplo, o comportamento do campo. Esse comportamento pode estar associado a parâmetros a serem inseridos, como *double* ou *int*, associados a características de um dispositivo ou à quantidade de caracteres que um usuário pode digitar em determinado campo.

Exemplo

Carlos, um profissional autônomo especialista em engenharia de *software*, atende a diversos clientes. Essa semana, um de seus clientes, a empresa StarBar, enfrentou alguns problemas com seu sistema de CRM. Até pouco tempo, a empresa só vendia seus produtos, etiquetas de código de barras, para empresas jurídicas, mas com o crescimento de vendas por pessoas físicas pela internet surgiu uma nova demanda. Com isso, seu sistema de CRM deixou de atender a determinados requisitos, como o campo de documentos (RG e CPF), nome social e redes sociais. A empresa StarBar solicitou então os serviços de Carlos, que, identificou que o cenário era típico de modificações em tempo de implantação. Por sua experiência, ele sabe que o sistema de CRM é desenvolvido sob uma linha-base e adaptado ao cliente de acordo com as regras de negócios específicas. Nesse caso, Carlos, por conhecer programação de sistemas CRM, fez as alterações de parâmetros necessárias para que o sistema atendesse aos novos requisitos e solucionou o problema do seu cliente.

Em geral, uma linha de produção de *software* beneficia as empresas ao otimizar processos de desenvolvimento de *software*, resultando em diminuição de custos, redução de tempo, criação de linhas para novos mercados, elevação da qualidade, etc., além da técnica de reúso aplicada ao sistema legado, que permite que componentes sejam modificados para se adequarem aos novos requisitos. Assim, é possível gerar soluções para diversas plataformas e

também se adaptar às necessidade de diferentes ambientes de implantação com base nas necessidades funcionais, bem como nos processos existentes específicos de cada negócio.

Ao longo do processo de uma linha de produção, modificações podem ser realizadas. Quando aplicadas nos componentes de uma aplicação legada, criando versões evolutivas da antiga aplicação, representam modificações em tempo de projeto. Já quando essas modificações são realizadas diretamente no cliente, ajustando-se às suas regras de negócio, representam modificações em tempo de implantação.

Referências

KEYES, J. *Software engineering handbook*. Boca Raton: Auerbach, 2003.

LECESSI, R. *Functional interfaces in Java*: fundamentals and examples. Kendall Park: Apress, 2019.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software*: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

REACT. *Introdução*. [*S. l.*]: Facebook, 2021. Disponível em: https://pt-br.reactjs.org/docs/getting-started.html. Acesso em: 7 set. 2021.

SOMMERVILLE, I. *Engenharia de software*. 10. ed. São Paulo: Pearson, 2018.

Leituras recomendadas

SCHACH, S. R. *Engenharia de software*: os paradigmas clássico & orientado a objetos. 7. ed. Porto Alegre: AMGH, 2010.

TSUI, F.; KARAM, O.; BERNAL, B. *Essentials of software engineering*. 3rd ed. Burlington: Jones & Bartlett Learning, 2014.

VILLELA, R. *Pro .NET framework with the base class library*: understanding the virtual execution system and the common type system. [Kendall Park]: Apress, 2019.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS