# ENGENHARIA DE SOFTWARE

**Aline Zanin**

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

# Conhecer os modelos tradicionais

## Objetivos de aprendizagem

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

- Reconhecer o modelo cascata, seu funcionamento, suas vantagens e suas desvantagens.
- Aplicar o modelo prototipação, seu funcionamento, suas vantagens e suas desvantagens.
- Caracterizar e identificar o modelo espiral, seu funcionamento, suas vantagens e suas desvantagens.

## Introdução

Os modelos clássicos foram criados entre os anos 1970 e 1990 por evoluções ocorridas no desenvolvimento de *software* para resolver problemas que representavam barreiras na criação de sistemas de alta qualidade. O modelo cascata foi um dos primeiros a serem propostos, referenciado em livros e amplamente conhecido até hoje. Após sua criação, diversos problemas foram apontados, levando ao surgimento de outros modelos, como o espiral e o prototipação. Ainda, houve a evolução dos modelos de ciclo de vida, possibilitando atualmente a utilização de diversos modelos. Para aprender sobre os modelos atuais, é muito importante entender os modelos tradicionais, identificar quais foram as evoluções que ocorreram e o motivo da presença de determinadas características nos modelos que utilizamos hoje.

Neste capítulo, você adquirirá conhecimentos fundamentais sobre os modelos de ciclo de vida tradicionais. Para isso, abordaremos os modelos cascata, espiral e prototipação, apresentando conceitos básicos sobre eles, além de suas vantagens e desvantagens.

# 1 Modelo cascata

Oficialmente, foi o primeiro modelo de ciclo de vida de desenvolvimento de *software*, motivo pelo qual é o mais conhecido entre os profissionais e estudantes da área de desenvolvimento de sistemas, sobretudo de engenharia de *software*. O modelo recebe o nome de cascata por ser um processo executado em sequência, sem que de uma etapa posterior seja possível retornar a uma etapa anterior — esse fluxo se assemelha ao do funcionamento de uma cascata, na qual o fluxo da água é contínuo para apenas uma direção.

Também chamado de ciclo de vida clássico, esse modelo sugere uma abordagem sequencial e sistemática para o desenvolvimento de *software*, começando com a especificação dos requisitos do cliente, avançando pelas fases de planejamento, modelagem, construção e disponibilização, e culminando no suporte contínuo do *software* concluído (Figura 1) (PRESSMAN 2011).

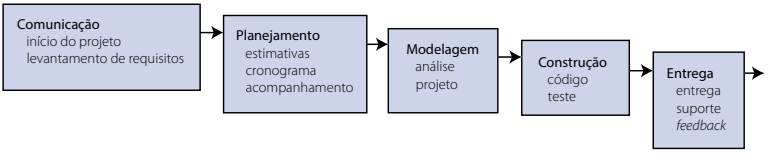


**Figura 1.** Modelo cascata.
***Fonte:*** Adaptada de Pressman (2011).

Empregamos o modelo cascata em casos nos quais os requisitos de um problema são bem compreendidos e razoavelmente estáveis, ou seja, quando o trabalho flui da comunicação à disponibilização de modo relativamente linear (PRESSMAN 2011).

Justamente por sua característica sequencial, esse modelo apresenta etapas muito bem definidas, o que facilita o gerenciamento dos projetos que o adotam. Isso acontece porque a cada etapa são gerados produtos entregáveis, o que possibilita que os gestores do projeto acompanhem claramente os resultados.

Outra vantagem do método cascata, também relacionada à separação do projeto em etapas, reside no fato de que os requisitos costumam estar bem definidos. O modelo não prevê alterações de requisitos no meio do processo, mas apenas uma definição detalhada no início que será implementada criteriosamente para ser entregue ao final do fluxo.

Contudo, embora a definição de requisitos no início do projeto permita um detalhamento maior, esse modelo desperta um problema clássico da engenharia de *software*: mudanças de requisitos, sejam por necessidades do cliente, necessidades tecnológicas ou viabilidade de mercado, não são bem recebidas e podem causar grandes transtornos para o projeto. Isso porque os requisitos nesse modelo são alterados apenas após a entrega da última etapa do modelo cascata, motivo pelo qual as mudanças podem ser significativas e demandar um alto tempo e custo.

Ainda, assim como os requisitos não são mutáveis, quando identificada uma falha na etapa imediatamente anterior à etapa corrente, o processo não permite que se retroceda a uma fase para realizar a correção, obrigando a seguir o fluxo até o final, uma característica que eventualmente se torna uma dificuldade ou até mesmo um empecilho para a execução real desse modelo.

Por fim, outra de suas desvantagens é que o cliente apenas visualizará o produto que está comprando e terá a oportunidade de ofertar *feedback* no último ciclo, faltando uma interação maior do cliente com a equipe de desenvolvimento e entre a própria equipe de uma fase para outra.

**Fique atento**

O modelo cascata considera as fases de análise e definição de requisitos, projeto de sistema e de *software*, implementação e teste de unidades, integração e teste de sistemas, operação e manutenção.

## 2 Modelo prototipação

Assemelha-se ao modelo cascata, considerando que também tem etapas bem definidas, mas apresenta uma peculiaridade como característica principal: tem em conta uma prototipação ao início do ciclo de vida. Essa prototipação é feita após um primeiro contato com o cliente e objetiva que todos os envolvidos consigam interagir com esse protótipo, tendo uma ideia de como o *software* funcionará para aprovar ou sugerir modificações.

O processo de prototipar antes do início da criação do sistema apresenta vantagens e desvantagens. A principal vantagem consiste no fato de que os requisitos se tornam mais visuais e intuitivos, uma vez que o profissional redigirá o documento de especificação de requisitos tendo o protótipo como base. Além disso, possibilita que o cliente tenha maior certeza de que expressou suas necessidades corretamente e que, por isso, receberá o *software* esperado.

Contudo, o protótipo pode ser considerado uma fonte de retrabalho e alto custo, uma vez que, caso o protótipo inicial não atenda às necessidades do cliente, precisará ser reconstruído, e, mesmo que seja aceito o primeiro protótipo, sua criação representa uma etapa extra, que envolve um trabalho de produção e validação. Além disso, por considerar um tempo de criação e validação do protótipo, o ciclo de vida do *software* como um todo pode demandar maior tempo para ser executado, processo visualizado na Figura 2.

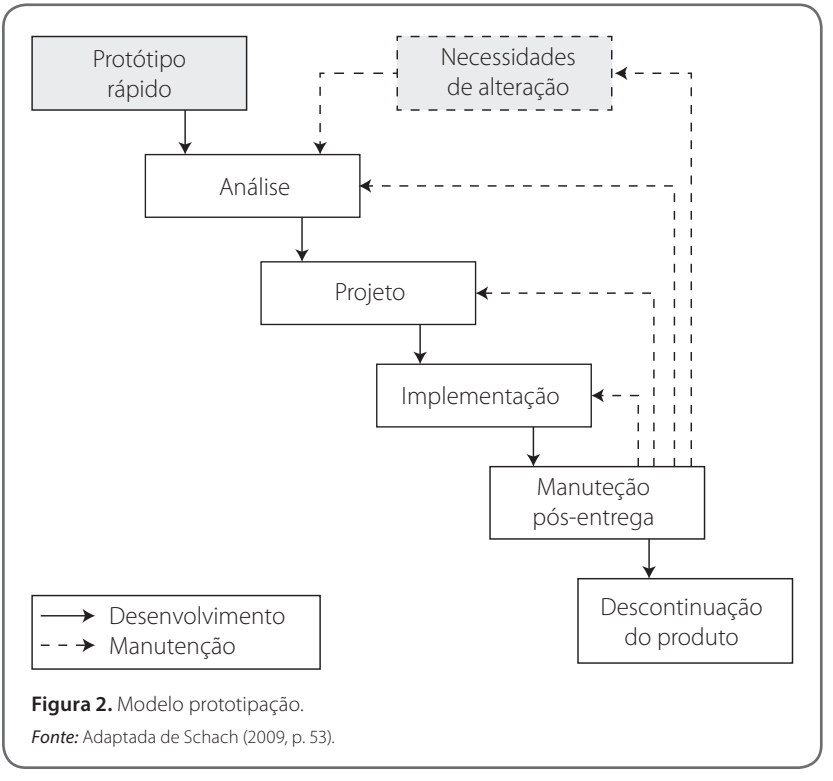


**Figura 2.** Modelo prototipação.

***Fonte:*** Adaptada de Schach (2009, p. 53).

## 3 Modelo espiral

Trata-se de um modelo que busca reunir características dos demais modelos de ciclos de vida tradicionais, considera também a criação de protótipos e a execução por fases bem definidas.

Sua principal diferença em relação ao modelo cascata é que o modelo espiral acrescenta ao cascata uma análise criteriosa de riscos ao início de cada etapa, beneficiando-se, assim, da principal característica do modelo prototipação, a criação de protótipos.

Pelo fato de unificar características dos demais modelos, o modelo espiral também tem vantagens e desvantagens quanto aos outros, pois apresenta requisitos e etapas bem definidos, mas com um alto custo de execução, por considerar diversas prototipações, uma para cada fase percorrida.

Criado por Barry Boehm em 1988, o modelo espiral é uma melhoria do modelo incremental e teve seu nome cunhado em virtude de sua representação, em que cada volta no espiral percorre todas as fases do processo de *software* (DIAS, 2019). De acordo com Pressman (2011) e Boehm e Hansen (2001), o modelo espiral de desenvolvimento é um gerador de modelos de processos dirigidos a riscos e utilizado para guiar a engenharia de sistemas com muito *software*, que ocorre de forma concorrente e tem vários envolvidos. Além disso, há duas características principais que o distinguem: a primeira consiste em uma estratégia cíclica voltada para ampliar, de modo incremental, o grau de definição e a implementação de um sistema, enquanto diminui o seu grau de risco; e a segunda reside no fato de dispor de uma série de marcos de pontos-âncora para garantir o comprometimento dos envolvidos quanto à busca de soluções de sistema mutuamente satisfatórias e viáveis. O modelo de ciclo de vida espiral (BOEHM, 1998 *apud* SCHACH, 2009) pode ser observado na Figura 3.

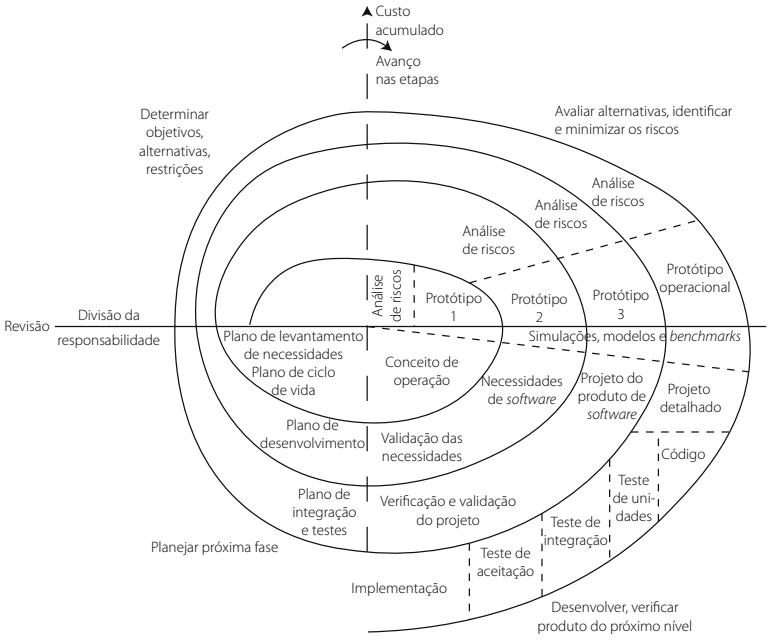


**Figura 3.** Modelo espiral de Boehm.

*Fonte:* Adaptada de Boehm (2000).

Na Figura 3, a dimensão radial representa o custo acumulado até o momento, e a dimensão angular o processo ao longo da espiral. Os ciclos correspondem às fases, em que uma fase se inicia (no quadrante superior esquerdo) concomitantemente à determinação de seus objetivos, com alternativas para atingi-los e restrições impostas sobre eles. Na sequência, essa estratégia é analisada sob o ponto de vista de riscos, que passam por tentativas de minimização, construindo-se, em alguns casos, protótipos. Se os riscos não puderem ser minimizados, o projeto pode ser interrompido imediatamente; contudo, se forem minimizados com sucesso, a próxima etapa de desenvolvimento é iniciada (quadrante direito inferior, que corresponde ao modelo cascata clássico). Por fim, os resultados são avaliados e se planeja a fase seguinte (SCHACH, 2009).

Pressman (2011) também aponta que, com o uso do modelo espiral, o *software* será desenvolvido em uma série de versões evolucionárias, que, nas primeiras, iterações poderão consistir em um modelo ou em um protótipo. Já nas iterações posteriores, são produzidas versões cada vez mais completas do sistema que passa pelo processo de engenharia. Ainda de acordo com o autor, o modelo espiral divide-se em um conjunto de atividades que representam um segmento do caminho espiral, no qual, assim que o processo evolucionário começa, a equipe de *software* realiza atividades indicadas por um circuito em torno da espiral, no sentido horário, iniciando pelo seu centro. Para tal, os riscos são levados em conta à medida que se realiza cada revolução. Por fim, o primeiro circuito em volta da espiral pode resultar no desenvolvimento de uma especificação de produto, enquanto as passagens subsequentes em torno da espiral podem ser usadas para desenvolver um protótipo e, então, progressivamente, versões cada vez mais sofisticadas do *software* (PRESSMAN, 2011).

**Exemplo**

A empresa Xpto tem trabalhado em um projeto com o objetivo desenvolver um sistema de um *e-commerce*.

- Caso opte por utilizar o modelo cascata, reunir-se-á com o cliente uma única vez, documentará os requisitos que o cliente solicitou e os apresentará para validação do cliente. Uma vez aceitos os requisitos, o *e-commerce* é criado e o cliente apenas o visualiza após todos os testes terem sido executados e o projeto encerrado.
- Caso se opte pelo modelo prototipação, antes de iniciar o desenvolvimento, os profissionais criarão as telas do *e-commerce* e as mostrarão para o cliente. O cliente deverá interagir como se estivesse usando o *software* real, para, então, aprovar o desenvolvimento ou sugerir alterações.
- No caso de utilizar o modelo espiral, será feito inicialmente o protótipo de uma tela de cadastro dos produtos, que será avaliada pelo cliente e, depois, desenvolvida. Em seguida, será feita uma tela para compras, também avaliada pelo cliente e, então, desenvolvida, e assim sucessivamente, prototipando cada parte do programa antes de ele ser desenvolvido. Além da prototipação, a cada etapa serão analisados os riscos possíveis em cada etapa, por exemplo, ter um estagiário que está se formando e que precisará deixar a equipe, uma mudança de versão de alguma tecnologia durante o processo, entre outros riscos previsíveis que, nesse modelo, são analisados previamente a cada ciclo.

## Referências

BOEHM, B. *Spiral development*: experience, principles, and refinements spiral development workshop. 2000. Disponível em: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5053. Acesso em: 08 jun. 2020.

BOEHM, B.; HANSEN, W. J. *The spiral model as a tool for evolutionary software acquisition*. 2001. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.642.8250&rep=rep1&type=pdf . Acesso em: 08 jun. 2020.

DIAS, R. P. *O modelo em espiral de Boehm*. 2019. Disponível em: https://medium.com/contexto-delimitado/o-modelo-em-espiral-de-boehm-ed1d85b7df. Acesso em: 08 jun. 2020.

PRESSMAN, R. S. *Engenharia de software*: uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011.

SCHACH, S. R. *Engenharia de software*: os paradigmas clássico e orientado a objetos. Porto Alegre: AMGH, 2009.

## 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*.

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