TRANS_
FORMAR
O PAÍS PELA
EDUCAÇÃO
É O QUE
NOS MOVE

ecossistema
ănima

Modelos, Métodos e Técnicas da Engenharia de Software

Aula 02 e Aula 03

Modelo de processo cascata. Modelo de processo de prototipação. Modelo de processo incremental. Modelo de processo espiral.

Prof. Luis Ybarra

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).

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.** (Morais, 2017).

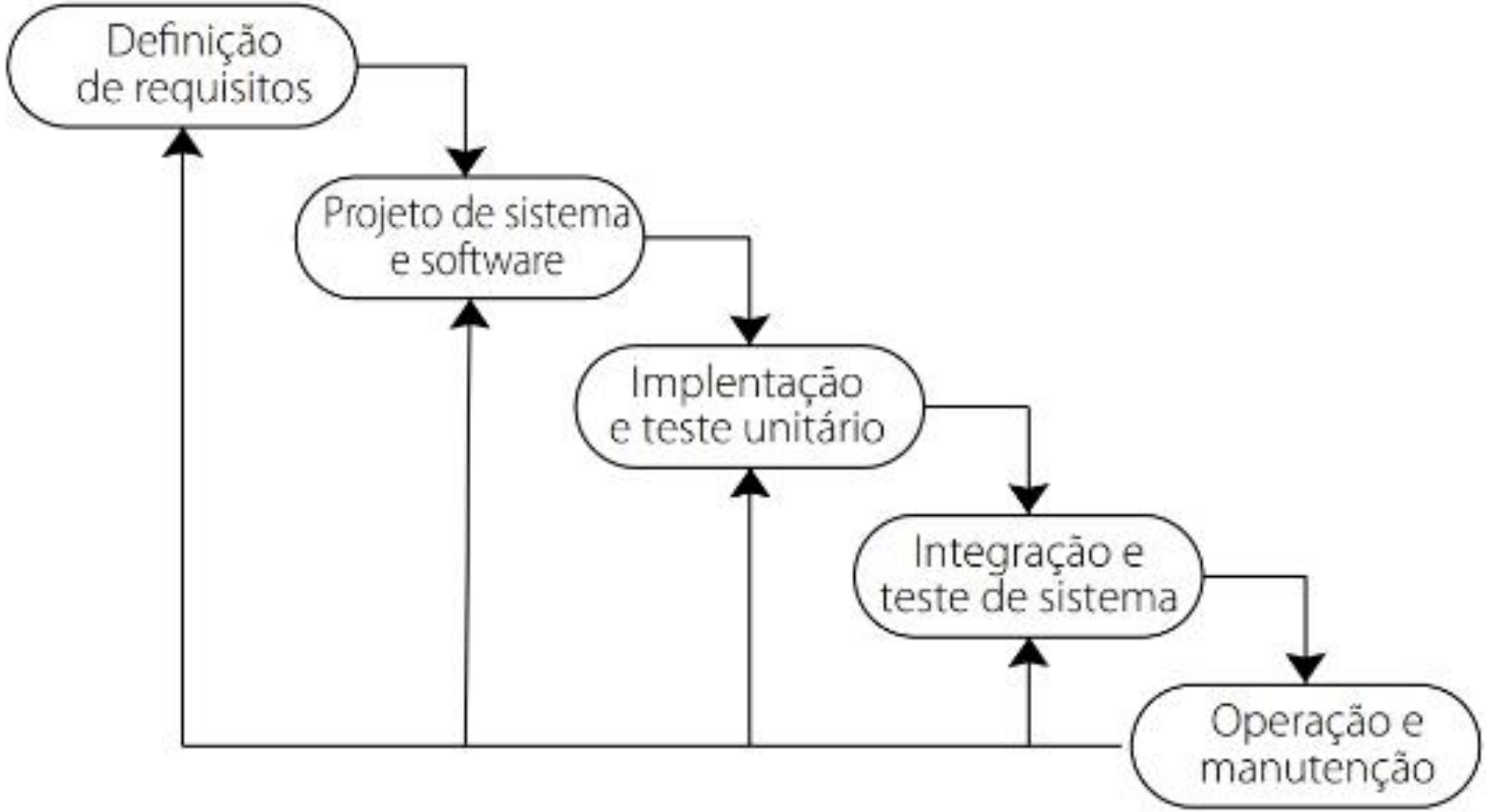


Figura 1. Modelo Cascata

Fonte: Sommerville (2011, p. 20).

UM MODELO DE PROCESSO GENÉRICO

O Processo foi definido como um conjunto de atividades de trabalho, ações e tarefas realizadas quando algum artefato de software deve ser criado.

Cada uma dessas atividades, ações e tarefas se aloca dentro de uma metodologia ou modelo que determina sua relação com o processo e uma com a outra.

Uma metodologia de processo genérica para engenharia de software estabelece cinco atividades metodológicas: **comunicação, planejamento, modelagem, construção e entrega.**

Além disso, um conjunto de atividades de apoio é aplicado ao longo do processo, como o acompanhamento e o controle do projeto, a administração de riscos, a garantia da qualidade, o gerenciamento das configurações, as revisões técnicas, entre outras.

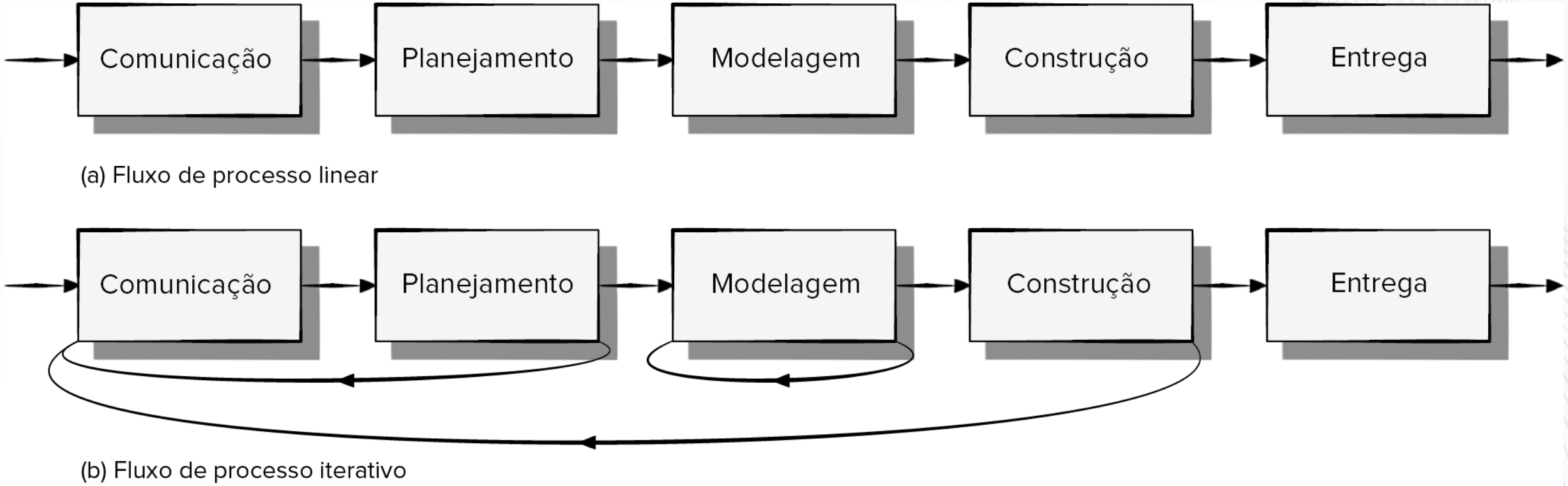
ă

Um aspecto importante do processo de software ainda não foi discutido.

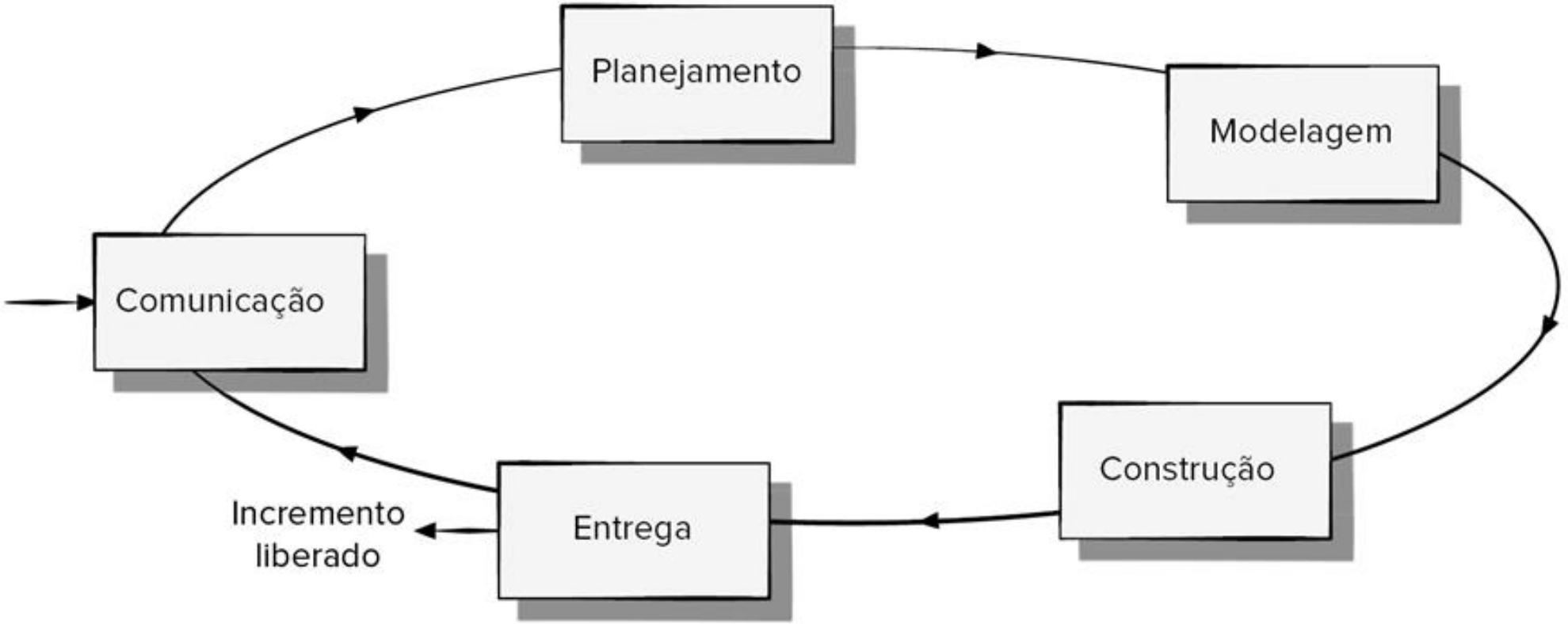
Este aspecto – chamado de fluxo de processo – descreve como são organizadas as atividades metodológicas, bem como as ações e tarefas que ocorrem dentro de cada atividade em relação à sequência e ao tempo, como ilustrado na Figura 2.2.

ă

Um fluxo de processo linear executa cada uma das cinco atividades metodológicas em sequência, começando com a comunicação e culminando com a entrega (Figura 2.2a).
Um fluxo de processo iterativo repete uma ou mais das atividades antes de prosseguir para a seguinte (Figura 2.2b).

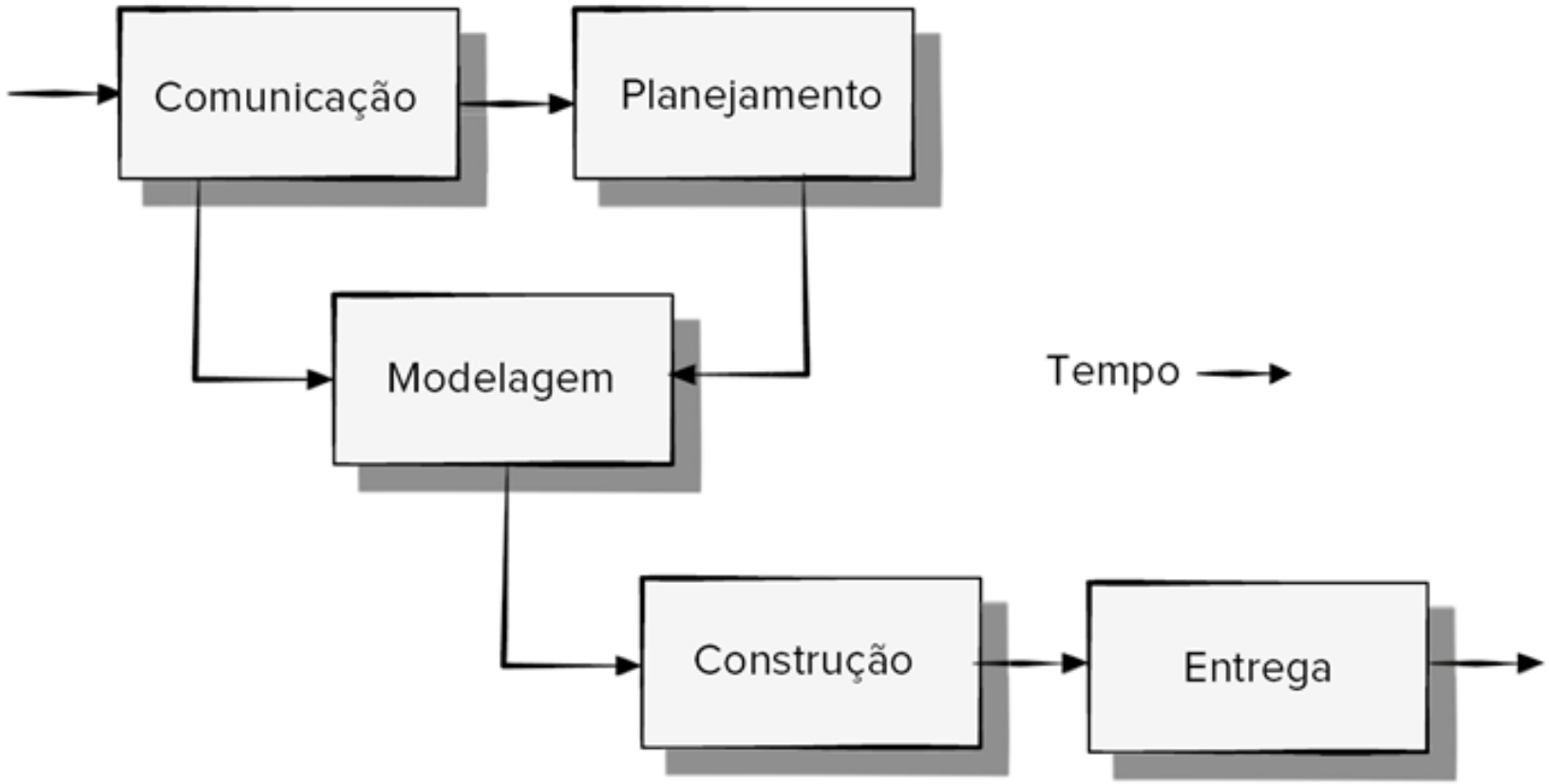


Um fluxo de processo evolucionário executa as atividades de forma “circular”. Cada volta pelas cinco atividades conduz a uma versão mais completa do software (Figura 2.2c).



(c) Fluxo de processo evolucionário

Um fluxo de processo paralelo (Figura 2.2d) executa uma ou mais atividades em paralelo com outras (p. ex., a modelagem para um aspecto do software poderia ser executada em paralelo com a construção de outro aspecto do software).



(d) Fluxo de processo paralelo

2.2 Definição de uma atividade metodológica

Uma equipe de software precisa de muito mais informações antes de poder executar qualquer uma das atividades como parte do processo de software.

Assim, enfrenta-se uma questão-chave: Quais ações são apropriadas para uma atividade metodológica, uma vez fornecidos a natureza do problema a ser solucionado, as características das pessoas que farão o trabalho e os envolvidos no projeto?

Para um pequeno projeto de software solicitado por uma única pessoa (em um local distante) com requisitos simples e objetivos, a **atividade de comunicação** pode se resumir a pouco mais de um telefonema ou e-mail para o envolvido.

Portanto, a única ação necessária é uma conversa telefônica, e as tarefas de trabalho (o conjunto de tarefas) que essa ação envolve são:

1. Contatar o envolvido via telefone.
2. Discutir os requisitos e gerar anotações.
3. Organizar as anotações em uma breve relação de requisitos, por escrito.
4. Enviar um e-mail para o envolvido para revisão e aprovação.

Se o projeto fosse consideravelmente mais complexo, com muitos envolvidos, cada qual com um conjunto de requisitos diferentes (por vezes conflitantes), a atividade de comunicação poderia ter seis ações distintas: concepção, levantamento, elaboração, negociação, especificação e validação.

Cada uma dessas ações de engenharia de software conteria muitas tarefas de trabalho e uma série de diferentes artefatos.

ă

2.5 Modelos de processo prescritivo

Chamamos esses processos de “prescritivos” porque prescrevem um conjunto de elementos de processo – atividades metodológicas, ações de engenharia de software, tarefas, artefatos, garantia da qualidade e mecanismos de controle de mudanças para cada projeto. Cada modelo de processo também prescreve um fluxo de processo (também denominado fluxo de trabalho) – ou seja, a forma pela qual os elementos do processo estão relacionados.

Modelo cascata

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).

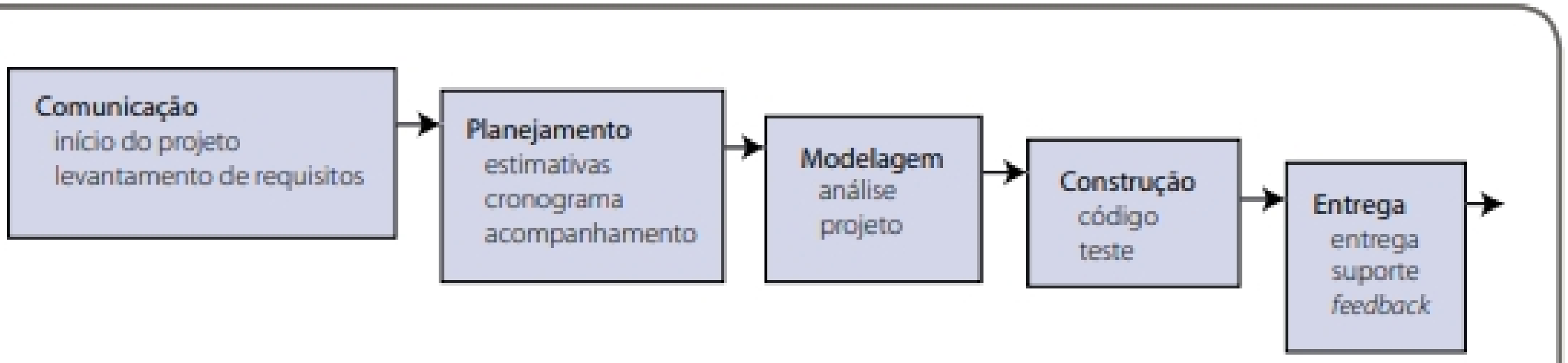


Figura 1. Modelo cascata.

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.

ă

Outra maneira do modelo cascata (modelo sequencial linear), 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 entrega, e culminando no suporte contínuo do software concluído (Figura 2.3).

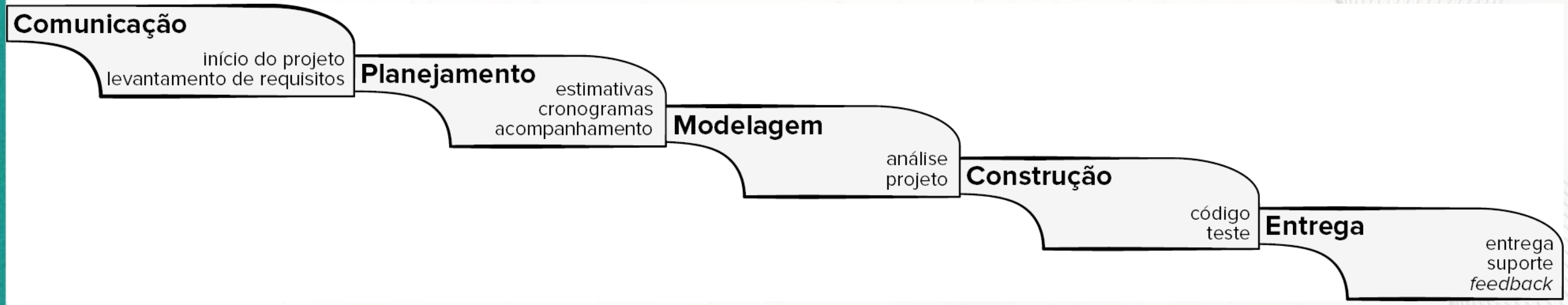


Figura 2.3 O modelo cascata.

Prática – Modelo Cascata

Elaborar em dupla, um projeto de sistemas, utilizando o Lucidchart

WWW.LucidChart.com

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.

ă

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

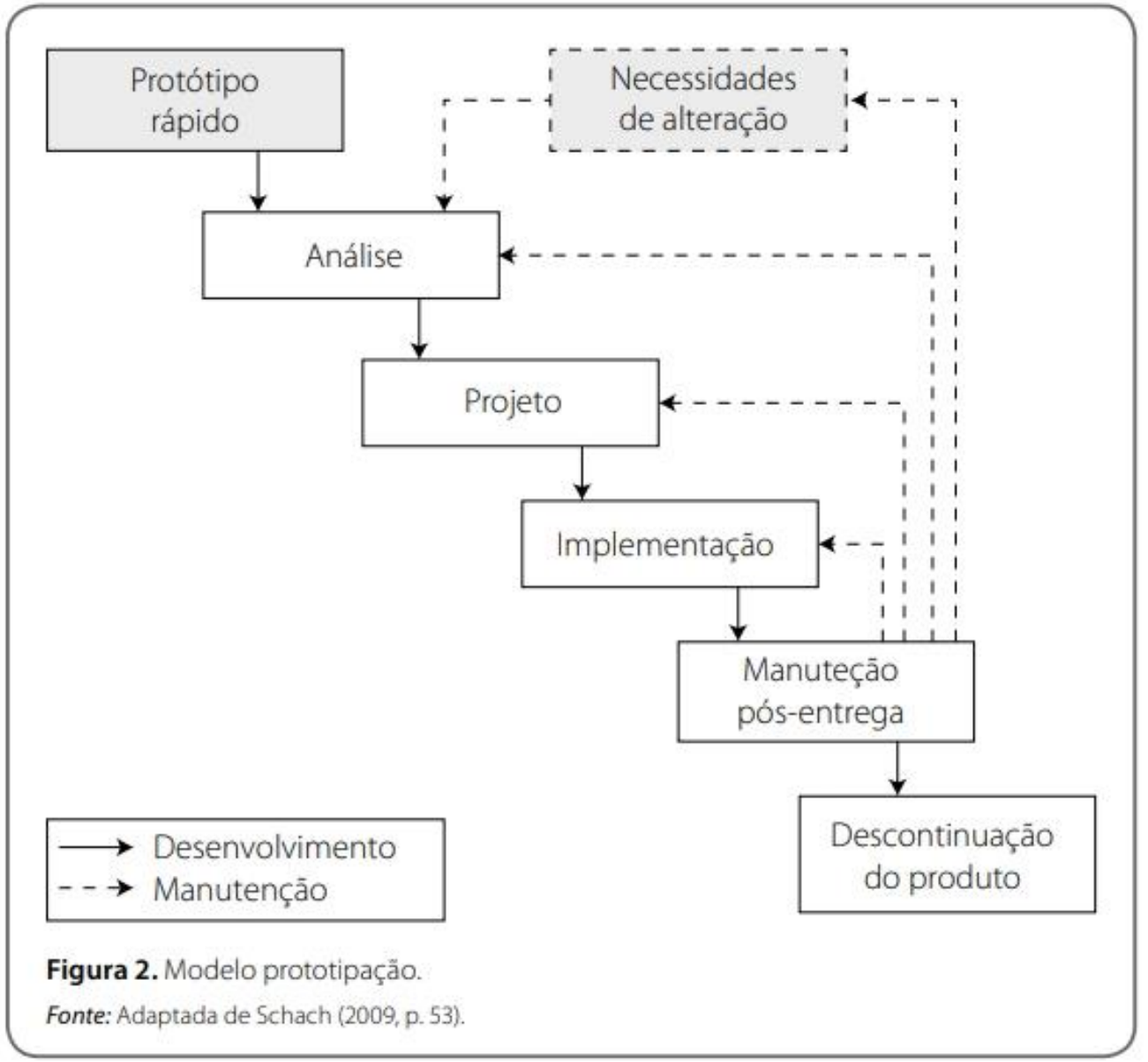


Figura 2. Modelo prototipação.

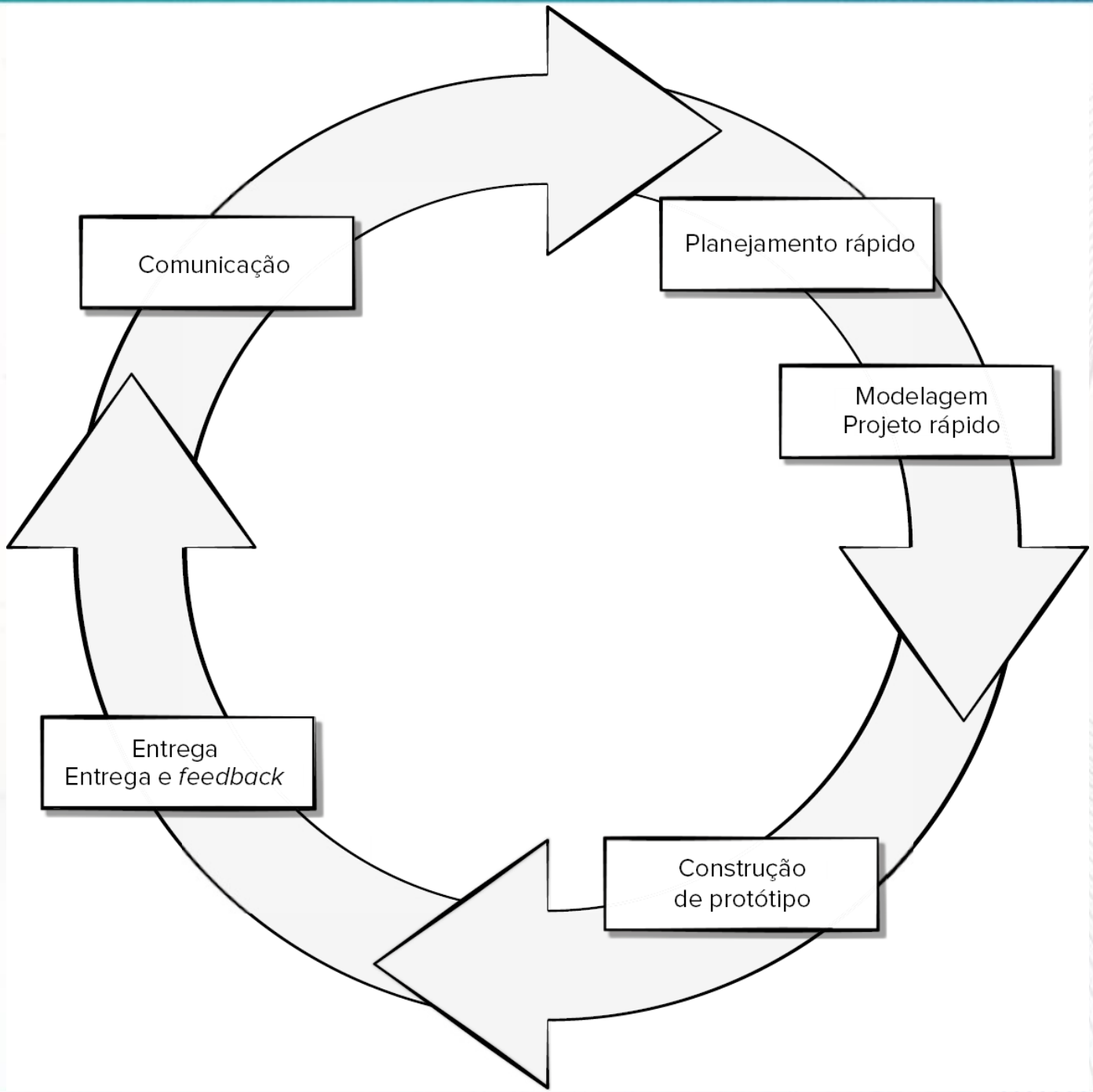
Fonte: Adaptada de Schach (2009, p. 53).

Outra forma de prototipação (Figura 2.4) começa com a comunicação. Faz-se uma reunião com os envolvidos para definir os objetivos gerais do software, identificar os requisitos já conhecidos e esquematizar quais áreas necessitam, obrigatoriamente, de uma definição mais ampla.

Uma iteração de prototipação é planejada rapidamente e ocorre a modelagem (na forma de um “projeto rápido”).

Um projeto rápido se concentra em uma representação dos aspectos do software que serão visíveis para os usuários (p. ex., o layout da interface com o usuário ou os formatos de exibição na tela).

O projeto rápido leva à construção de um protótipo. O protótipo é entregue e avaliado pelos envolvidos, os quais fornecem feedback que é usado para refinar ainda mais os requisitos.
A iteração ocorre conforme se ajusta o protótipo às necessidades de vários envolvidos e, ao mesmo tempo, possibilita a melhor compreensão das necessidades que devem ser atendidas.



Prática - Modelo Prototipação

Elaborar em dupla, um projeto de sistemas reestruturado, utilizando o Lucidchart

WWW.LucidChart.com

Modelos, Métodos e Técnicas da Engenharia de Software

Aula 03

Modelo de processo espiral. Modelo de processo incremental.

Prof. Luis Ybarra

Modelo de processo Espiral e/ou Evolucionário

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.

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.

O modelo espiral é dividido em um conjunto de atividades metodológicas definidas pela equipe de engenharia de software.

Cada uma dessas atividades representa um segmento do caminho espiral ilustrado na Figura 2.5.

Assim que esse processo evolucionário começa, a equipe de software realiza atividades indicadas por um circuito em torno da espiral, no sentido horário, começando pelo seu centro.

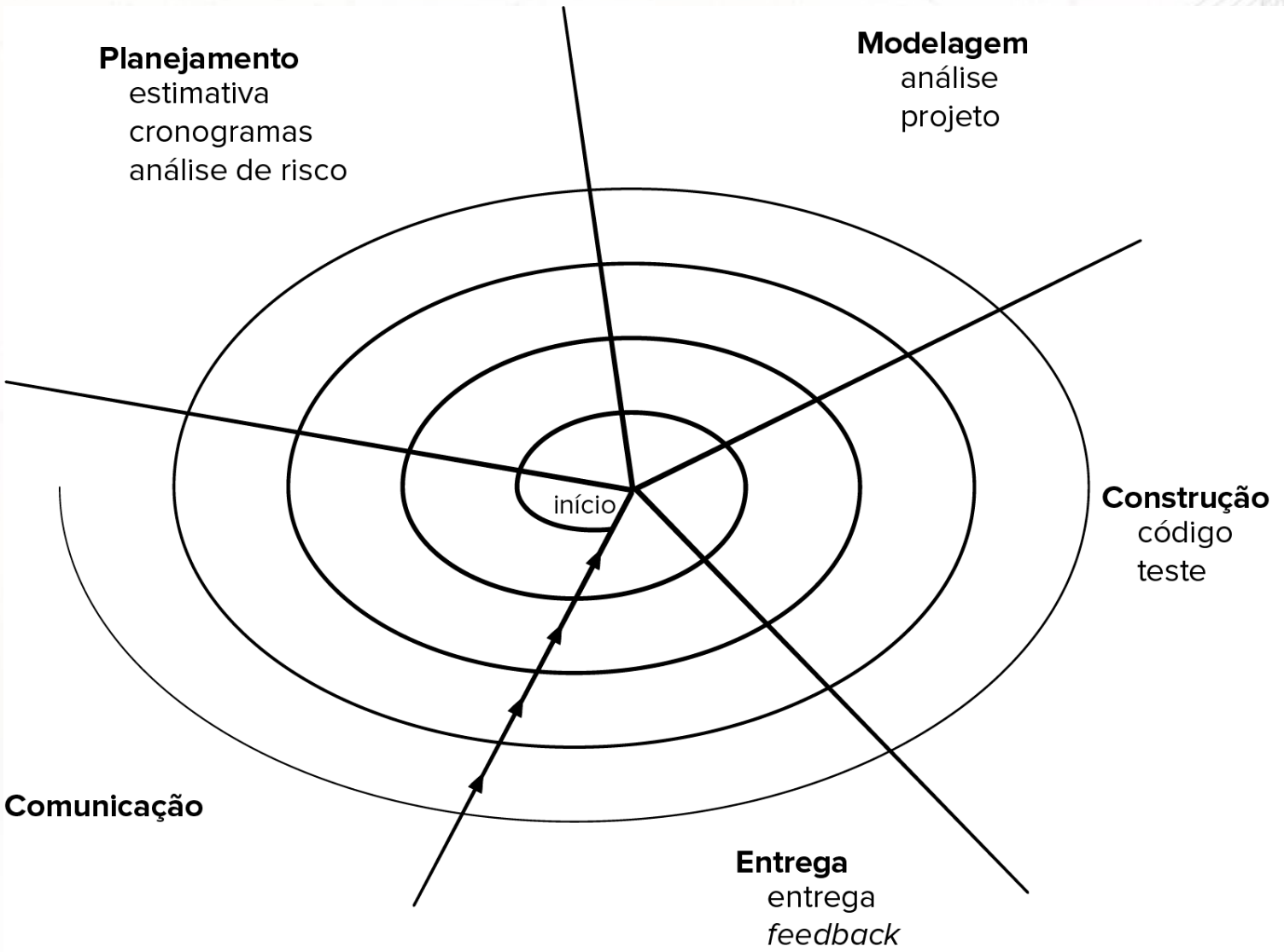


Figura 2.5 Modelo espiral típico.

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.

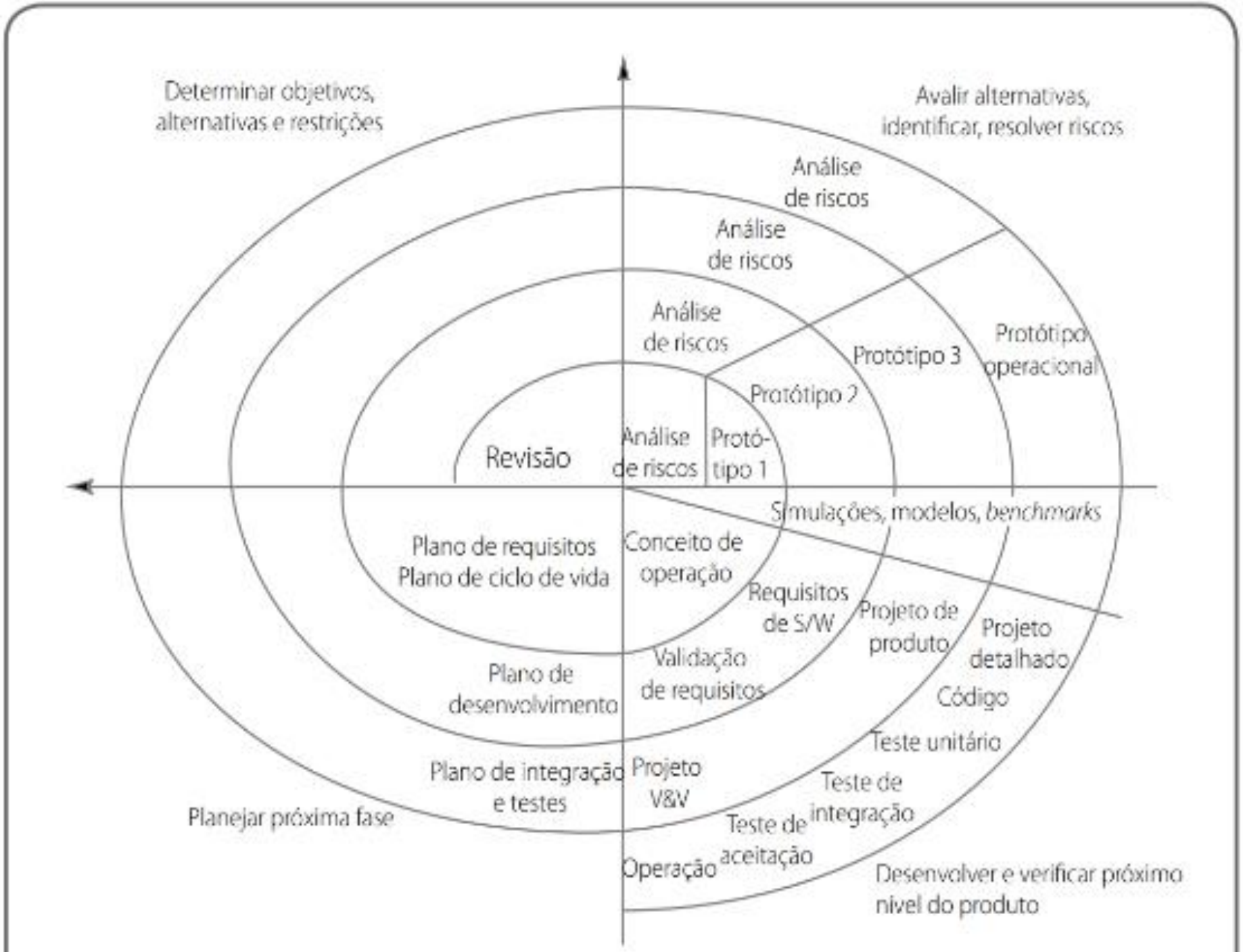


Figura 3. Modelo Espiral.

Fonte: Sommerville (2011, p. 33).

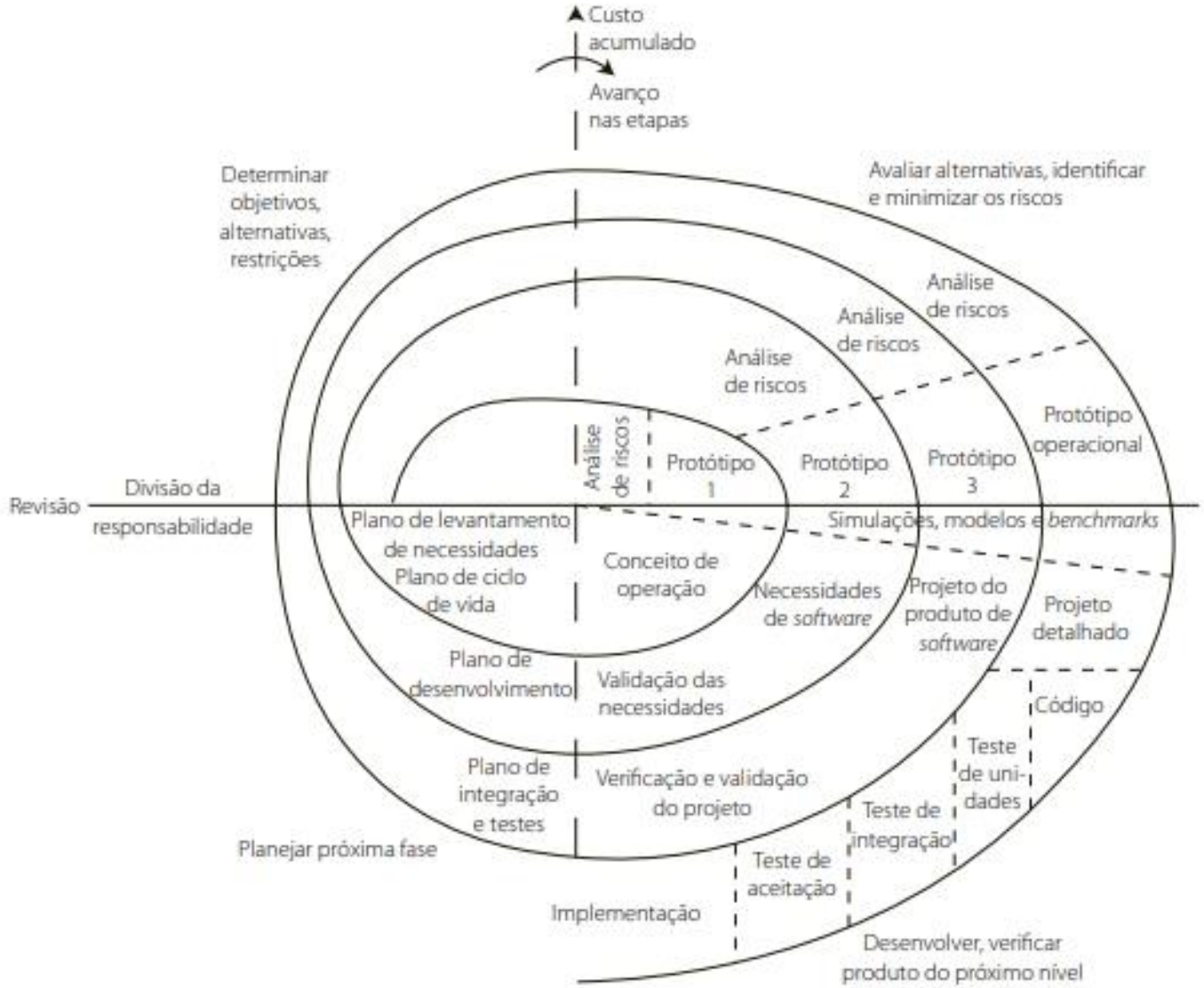


Figura 3. Modelo espiral de Boehm.

Fonte: Adaptada de Boehm (2000).

Prática – Modelo Espiral

Elaborar em dupla, um projeto de sistemas, utilizando o Lucidchart

WWW.LucidChart.com

Modelo de processo incremental.

O modelo incremental considera diversas entregas parciais do produto antes de uma entrega final, sendo esta a sua principal diferença em relação aos modelos prototipação e cascata.

Nesses modelos, o ciclo de desenvolvimento é contínuo e o cliente recebe um entregável do produto solicitado apenas quando o todo for concluído, ou seja, quando percorridas as seguintes etapas: análise e definição de requisitos, projeto de sistema e de software, implementação e teste de unidades, integração e teste de sistemas, operação e manutenção e, ainda, a prototipação no modelo de prototipação.

Já no modelo incremental, são intercaladas as atividades de especificação, desenvolvimento e validação.

Neste sentido, o sistema é desenvolvido como uma série de versões (incrementos), de maneira que cada versão adiciona funcionalidade à anterior (SOMMERVILE, 2011).

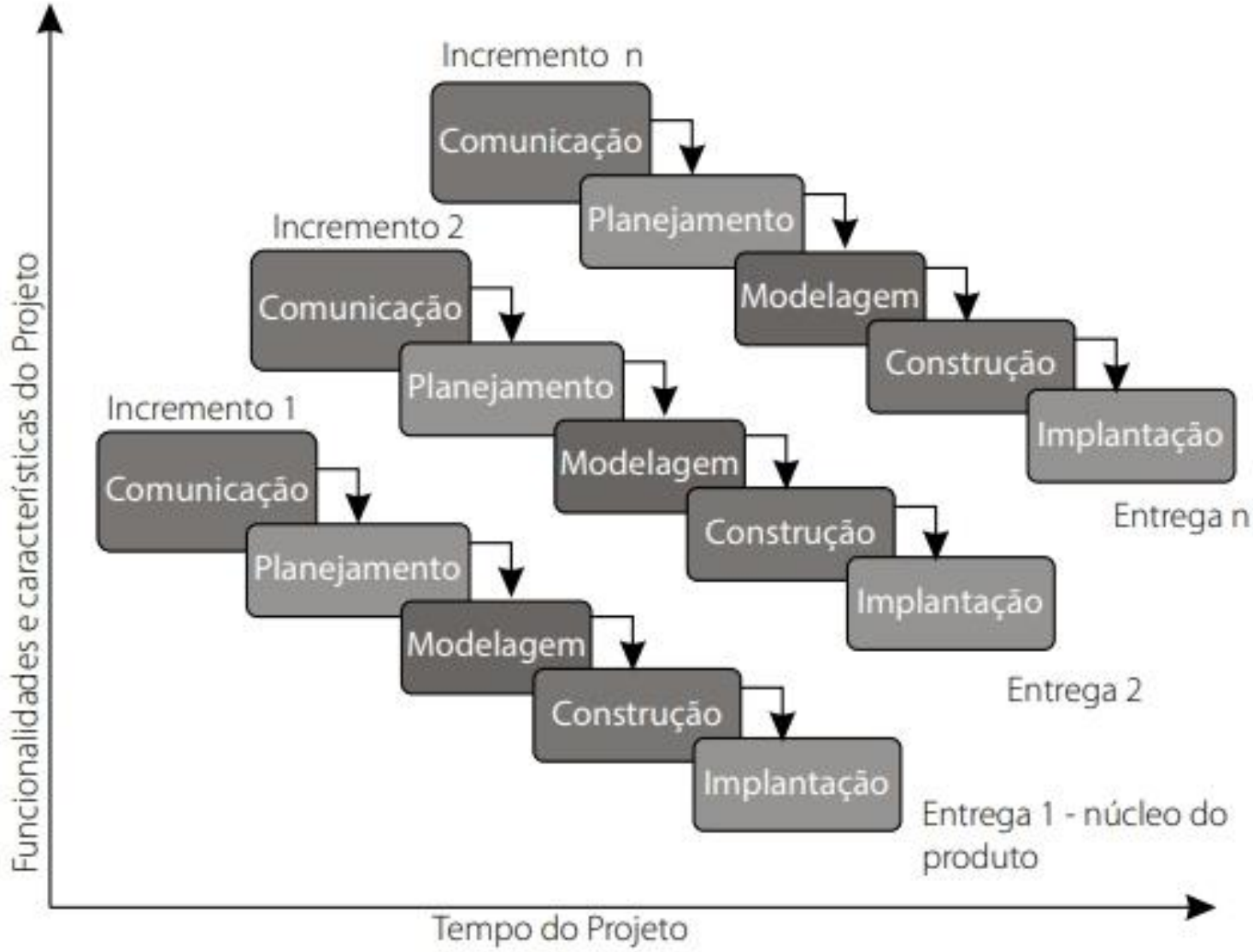


Figura 1. Exemplo de incrementos de funcionalidades.

Fonte: Nepomuceno (2012).

Segundo Morais (2017) o desenvolvimento **iterativo e incremental**, deste modelo, implementa-se pequenas partes entregáveis do software para que o cliente tenha um feedback mais rápido sobre o produto que está sendo desenvolvimento.

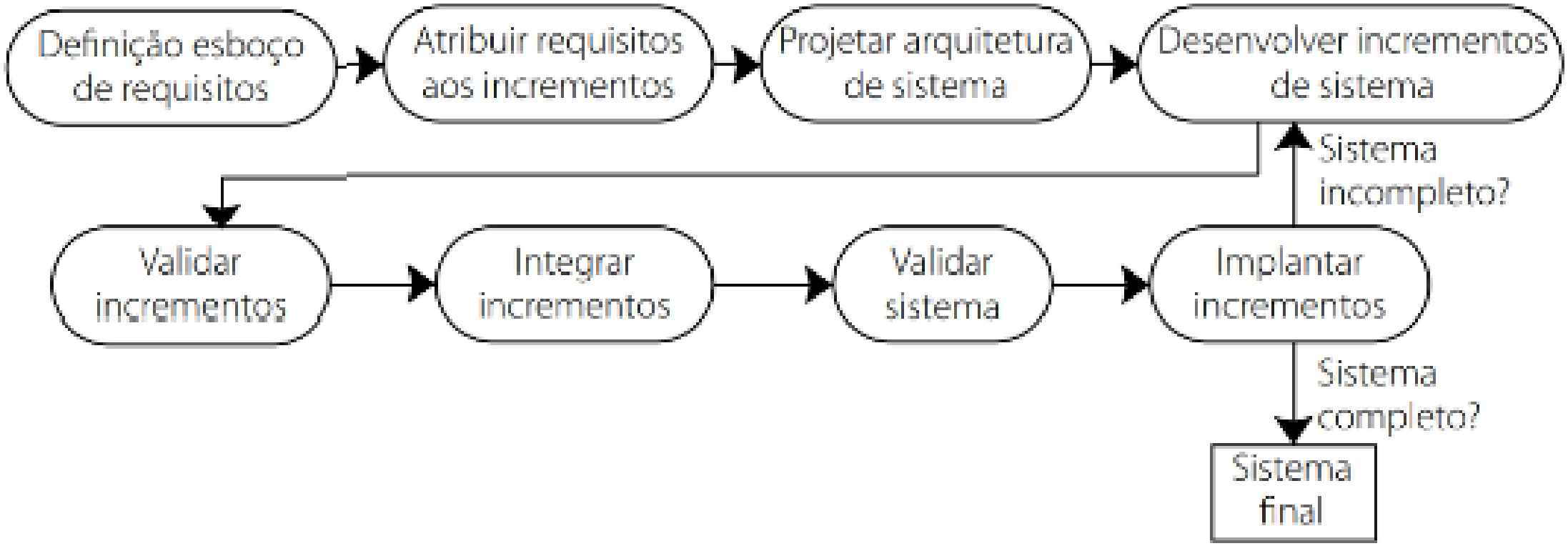


Figura 2. Entrega Incremental.

Fonte: Sommerville (2011, p. 31).

Identificar os incrementos

Como descrito anteriormente, o princípio fundamental do modelo incremental é a presença de ciclos de incrementos.

Pode-se dizer que o modelo incremental une os modelos cascata e prototipação justamente por reunir características de ambos. Desta forma, o princípio básico para a identificação de um incremento neste modelo é analisar as etapas do processo de desenvolvimento que estão sendo executadas.

Um incremento sempre inicia no levantamento das necessidades ou dos requisitos, sendo que, se estiver sendo executado o início do processo de desenvolvimento, esse incremento irá iniciar pela conversa com o cliente e pela identificação das necessidades dele.

Já no caso de um incremento que complementa ou é realizado para corrigir algo no software que está em desenvolvimento, inicia-se analisando aquilo que já foi programado e comparando com o que foi solicitado pelo cliente.

Em um processo ideal, o cliente participa do processo, analisando cada entrega ao final de um incremento e especificando o que espera do próximo incremento.

Vantagens e desvantagens do modelo incremental

A utilização deste modelo apresenta algumas **vantagens** para as empresas que optam por utilizá-lo.

Todas as vantagens estão diretamente relacionadas com a estrutura de trabalho por incrementos.

A seguir, listamos algumas dessas vantagens (IFSC, 2006):

- Redução dos custos com manutenção do sistema: esta vantagem se manifesta uma vez que, quando acontecem problemas no desenvolvimento por erros técnicos ou por não entendimento dos requisitos do cliente, os problemas serão rapidamente identificados, dado que cada ciclo é executado de forma curta e completa.

- Melhor controle de cronograma: por estarem executando pequenos ciclos completos de desenvolvimento, as equipes têm maior certeza de que, ao chegar ao último ciclo o programa, atenderão o esperado pelo cliente e, assim, evitam problemas de cronograma ocasionados por mudanças ou correções inesperadas.
- Maior probabilidade de atendimento dos requisitos do cliente: por estar realizando entregas contínuas e o cliente recebendo pequenas partes do produto solicitado, o cliente consegue ter uma melhor visão do produto que está sendo desenvolvido e, com isso, tem oportunidade de reportar quando os requisitos não estão sendo atendidos, desta forma, recebendo, ao final, um produto dentro do esperado.

ă

Porém, como todos os processos, este modelo tem algumas **desvantagens**, dentre elas (IFSC, 2006):

- dificuldade de gerenciamento: isso ocorre porque as fases de do ciclo podem estar ocorrendo de forma simultânea.
- para que o projeto tenha sucesso, o cliente precisa estar disposto a prover feedbacks constantes.
- arquitetura do projeto precisa ser bem estruturada para que possa receber os incrementos quando solicitado.
- cliente precisa estar ciente de que é um processo incremental e que não estará recebendo, nas primeiras entregas, o software final. Por isso, pode não estar completo.

Prática – Modelo incremental

Elaborar em dupla, um projeto de sistemas, utilizando o Lucidchart

WWW.LucidChart.com

BIBLIOGRAFIA

Pressman, Roger S. Engenharia de software : uma abordagem profissional [recurso eletrônico] / Roger S. Pressman, Bruce R. Maxim ; [tradução: João Eduardo Nóbrega Tortello ; revisão técnica: Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade]. – 8. ed. – Porto Alegre : AMGH, 2016.

Morais, Izabelly Soares de. Engenharia de software [recurso eletrônico] / Izabelly Soares de Morais, Aline Zanin ; revisão técnica : Jeferson Faleiro Leon. – Porto Alegre : SAGAH, 2017.

PRESSMAN, Roger; MAXIM, Bruce. Engenharia de Software. Uma abordagem profissional. 8a. Ed. Bookman, 2016. https://integrada.minhabiblioteca.com.br/#/books/9788580555349/cfi/3!/4/2@100:0.00

SOMMERVILLE, Ian. Engenharia de Software. 9. ed. São Paulo: Pearson Prentice Hall, 2011. https://bv4.digitalpages.com.br/?term=engenharia%2520de%2520software&searchpage=1&filtro=todos&from=busca&page=_14&section=0#/legacy/276

LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e desenvolvimento iterativo. 3. ed Porto Alegre: Bookman, 2007. https://integrada.minhabiblioteca.com.br/#/books/9788577800476/cfi/0!/4/2@100:0.00

IFSC. Ciclo de Vida Iterativo e Incremental. Wiki Instituto Federal de Santa Catarina, São José, out. 2006. Disponível em: <https://wiki.sj.ifsc.edu.br/wiki/index.php/Ci-clo_de_Vida_Iterativo_e_Incremental>. Acesso em: 31 ago. 2017.