



# Projetos Ágeis e Análise de Sistemas

# Projetos Ágeis e Análise de Sistemas

**Autora:** Juliana Schiavetto Dauricio

**Como citar este documento:** DAURICIO, Juliana S. *Projetos Ágeis e Análise de Sistemas*. Valinhos: 2016.

---

## Sumário

Apresentação da Disciplina	03
<b>Unidade 1:</b> Introdução ao Gerenciamento de Projetos	05
Assista suas aulas	24
<b>Unidade 2:</b> Planejamento de Projetos	33
Assista suas aulas	54
<b>Unidade 3:</b> Métodos Ágeis	63
Assista suas aulas	86
<b>Unidade 4:</b> Paradigma Orientado a Objetos: Histórico, Características e Conceitos	96
Assista suas aulas	119
<b>Unidade 5:</b> A Linguagem de Modelagem Unificada – Unified Modeling Language (UML): Histórico e Visão Geral das Técnicas de Modelagem. IFML	129
Assista suas aulas	147



# Projetos Ágeis e Análise de Sistemas

**Autora:** Juliana Schiavetto Dauricio

**Como citar este documento:** DAURICIO, Juliana S. *Projetos Ágeis e Análise de Sistemas*. Valinhos: 2016.

---

## Sumário

<b>Unidade 6:</b> Modelo de Use Cases: Diagrama de Use Cases e Documentação de Use Cases.	157
Especificação das Técnicas de Modelagem com Ferramenta Case	
Assista suas aulas	175
<b>Unidade 7:</b> Diagrama de Classes e de Sequência	186
Assista suas aulas	205
<b>Unidade 8:</b> Diagrama de Estado Comportamental	218
Assista suas aulas	235



## Apresentação da Disciplina

Olá, aluno! Bem-vindo à disciplina de Projetos Ágeis e Análise de Sistemas, do curso de pós-graduação em Tecnologias para Aplicações Web. Neste curso, você conhecerá como trabalhar sob o ponto de vista da gestão de projetos para o desenvolvimento de software e de que forma isso auxilia na análise de sistemas a considerar todas as variáveis que podem estar envolvidas, quais são as metodologias de desenvolvimento ágil mais conhecidas e utilizadas no mercado e, ainda, de que forma podem ser aplicadas. Além desses temas, você ainda será apresentado ao paradigma de programação orientada a objetos e quais ferramentas podem auxiliar na análise de sistemas a partir das premissas, tanto da gestão de projetos quanto da análise orientada a objetos.

Nesse contexto, aprenderá uma ferramenta que auxilie na modelagem de acordo com o padrão UML para desenvolver os diagramas de caso de uso, de sequência e de estado. Mãos à obra e bons estudos!



# Unidade 1

## Introdução ao Gerenciamento de Projetos

---

### Objetivos

Com esta disciplina, temos os objetivos de:

1. Apresentar ao aluno as premissas do gerenciamento de projetos para o desenvolvimento de softwares.
2. Possibilitar ao aluno conhecer e aplicar as metodologias de desenvolvimento ágil.
3. Incentivar que o aluno aprimore suas técnicas para melhorar os projetos e, também, o ritmo e integração da equipe, desde o início até o encerramento do projeto de desenvolvimento de software.



## Introdução


Vamos com esses estudos compreender, primeiramente, o que é um projeto, o que é um projeto ágil e de que forma se correlacionam com a análise de sistemas. Para tal, precisamos ter uma visão mais ampla: podemos começar com a compreensão do que é um projeto.

As boas práticas de gestão de projetos são disseminadas pelo Instituto de Gerenciamento de Projetos, conhecido como PMI (Project Management Institute), fundado em 1969 na Filadélfia – EUA, organizado juridicamente sob o regime de instituição sem fins lucrativos. Tais práticas são divulgadas através de um Guia do Conhecimento em Gerência de Projetos: o PMBOK (Project Management Body of Knowledge). Esse guia contém todas as

premissas que podem ser consideradas desde o início até o encerramento de um projeto. Atualmente, encontra-se em sua 5ª edição. Então, vamos lá! Mas, afinal, o que é um projeto?

Entende-se por projeto, de acordo com o PMBOK (2013), todo empreendimento de esforço e tempo que tem delimitados o seu início e fim, marcados por fases de gerenciamento que permitem enxergar e trabalhar cada uma das atividades do projeto, de forma a aumentar a qualidade da entrega, seja um produto, serviço, uma melhoria de processo, ou, ainda, um resultado como em um projeto de pesquisa ou outro documento referente ao projeto.





Nesse contexto, temos também que compreender o que vem a ser um projeto ágil, o que nos indica que com algumas técnicas, que conheceremos no decorrer de nossos estudos, há a aceleração no ritmo de desenvolvimento do projeto. Isso equivale a dizer que o cronograma conta com atividades sendo desenvolvidas simultaneamente por equipes que buscam manter maior proximidade com os envolvidos no projeto (clientes, equipe de projetos, etc.) e com o solicitante, respectivamente, conhecidos como *stakeholders* e *sponsor*.



SOTILLE, Mauro. Processos\_PMBOK\_4a\_Mauro\_Sotille.pdf. Disponível em: <[http://www.pmtech.com.br/artigos/Processos\\_PMBOK\\_4a\\_Mauro\\_Sotille.pdf](http://www.pmtech.com.br/artigos/Processos_PMBOK_4a_Mauro_Sotille.pdf)>. Acesso em: 13 abr. 2016. Analise as diferenças dos processos da 4ª e 5ª edição que Sotille disponibiliza.



Conheça uma das definições de projeto ágil que Sommerville (2011, p. 39) apresenta:



Os processos de desenvolvimento rápido de software são conhecidos para produzir rapidamente, softwares úteis. O software não é desenvolvido como uma única unidade, mas como uma série de incrementos – cada incremento inclui uma nova funcionalidade do sistema.

Os modelos de desenvolvimento ágil foram criados a partir da necessidade de integrar as equipes de projetos, ampliar a comunicação entre elas e o cliente, reduzir o tempo de execução do projeto, diminuir a incidência de falhas e, ainda, facilitar as etapas de planejamento e análise do sistema.


A partir de agora, quando ouvir análise de sistema, você já pode associar o conceito ao planejamento de todas as atividades que deverão ser consideradas para o desenvolvimento do software; em outras palavras, a análise de sistemas procura responder às necessidades dos clientes, de quem efetivamente usará o sistema que será disponibilizado. Entender todas as solicitações do usuário, que atendam às estratégias que precisam ser embutidas nos negócios e controladas por produtos de software, é uma tarefa árdua que conta com a [expertise](#) das áreas de projetos e análise de sistemas.



Não se esqueça de que o gerenciamento de projetos, para que seja aplicado a produtos de software, sempre trará consigo características que a Engenharia de Software considera essenciais e que devem ser os atributos de um software. Observe o quadro abaixo com a descrição de cada uma delas:

Quadro 1 – Atributos essenciais de um bom software

Características do produto de software	Descrição
Manutenibilidade	O software deve ser escrito de forma que possa evoluir para atender às necessidades dos clientes. Esse é um atributo crítico, porque a mudança de software é um requisito inevitável de um ambiente de negócios em mudança.
Confiança e proteção	A confiança do software inclui uma série de características como confiabilidade, proteção e segurança. Um software confiável não deve causar prejuízos físicos ou econômicos no caso de falha de sistema. Usuários maliciosos não devem ser capazes de acessar ou prejudicar o sistema.



Eficiência	O software não deve desperdiçar recursos do sistema, como memória e ciclos do processador. Portanto, eficiência inclui capacidade de resposta, tempo de processamento, uso de memória etc.
Aceitabilidade	O software deve ser aceitável para o tipo de usuário para o qual foi projetado. Isso significa que deve ser compreensível, usável e compatível com outros sistemas usados por ele.

Fonte: Sommerville, 2011, p. 5.

Finalmente, após conhecer alguns elementos novos e relembrar o que é um bom software de acordo com a Engenharia de Software, você já pode seguir em frente com a leitura de seu material e descobrir a importância de cada um desses elementos: gestão de projetos, projetos ágeis e análise de sistemas.

## Para saber mais

Recorde-se dos principais aspectos da Engenharia de Software, através da consulta aos capítulos 1 e 2 do livro de Sommerville, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

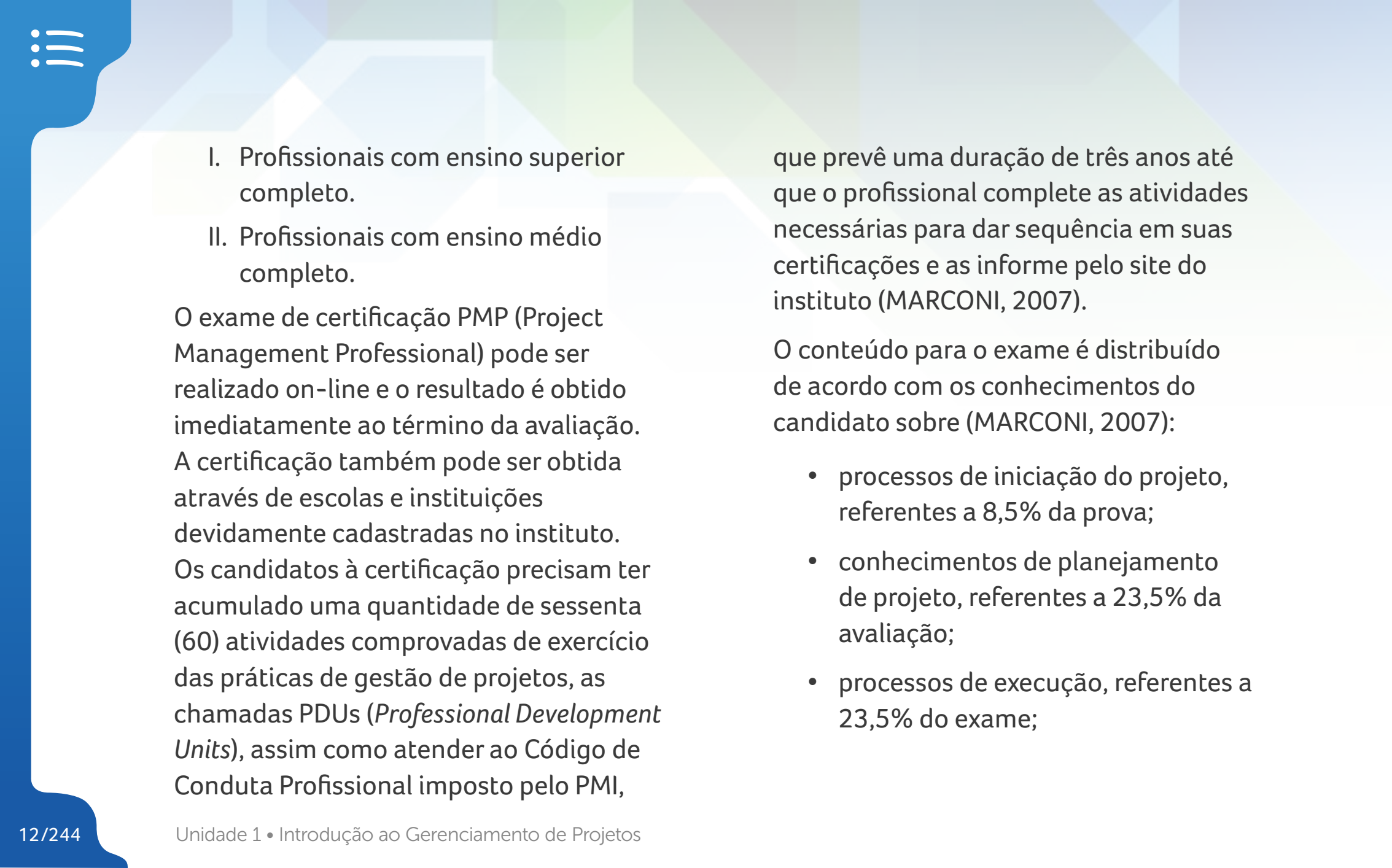
### 1. PMI (Project Management Institute)

O Instituto de Gerenciamento de Projetos, como já mencionado anteriormente, é uma instituição sem fins lucrativos, que é mantida por associados interessados em aprender e compartilhar suas experiências em gestão de projetos e estão presentes no mundo todo. A missão do PMI é

“fomentar o profissionalismo e a ética em gerenciamento de projetos” (MARCONI, 2007, p. 196).

A fim de capacitar pessoas que queiram exercer atividades profissionais na área de gestão de projetos, o PMI oferece certificações que comprovam que essas pessoas aprenderam e aplicam as boas práticas da área, conforme as prerrogativas do guia, o PMBOK.

Desde 1984, o PMI oferece aos associados um programa de certificação que se tornou bastante conhecido pelo seu padrão e o rigor das avaliações para obter o título. Estão divididas em duas categorias:

- 
- I. Profissionais com ensino superior completo.
  - II. Profissionais com ensino médio completo.

O exame de certificação PMP (Project Management Professional) pode ser realizado on-line e o resultado é obtido imediatamente ao término da avaliação. A certificação também pode ser obtida através de escolas e instituições devidamente cadastradas no instituto. Os candidatos à certificação precisam ter acumulado uma quantidade de sessenta (60) atividades comprovadas de exercício das práticas de gestão de projetos, as chamadas PDUs (*Professional Development Units*), assim como atender ao Código de Conduta Profissional imposto pelo PMI,

que prevê uma duração de três anos até que o profissional complete as atividades necessárias para dar sequência em suas certificações e as informe pelo site do instituto (MARCONI, 2007).

O conteúdo para o exame é distribuído de acordo com os conhecimentos do candidato sobre (MARCONI, 2007):

- processos de iniciação do projeto, referentes a 8,5% da prova;
- conhecimentos de planejamento de projeto, referentes a 23,5% da avaliação;
- processos de execução, referentes a 23,5% do exame;

- processos de monitoramento e controle, referentes a 23% da prova;
- encerramento e responsabilidade profissional, referentes a 14,5% da prova de certificação.

PMP, é a certificação mais comum entre os candidatos que desejam obter uma certificação do PMI. Já para o desenvolvimento ágil, a certificação é a conhecida no mercado de desenvolvimento de software como: PMI-ACP (*Agile Certified Practitioner*).

Nesse caso, o exame de certificação é composto por 120 questões que devem ser respondidas em três horas, de acordo com dados da revista *Mundo Project Management* (2012). O candidato precisa

ter conhecimento de duas grandes áreas: das “*ferramentas e técnicas ágeis, e conhecimento e habilidades ágeis*” (MUNDO PROJECT MANAGEMENT, 2012, p. 55).

Link



Acesse o site do PMI e leia os estudos de caso disponíveis, a fim de aprender sobre as suas certificações e como as boas práticas da gestão de projetos vêm sendo aplicadas. Disponível em: <[www.pmi.org](http://www.pmi.org)>. Acesso em: 19 abr. 2016.

## 2. Gerenciamento De Projetos

Agora que você já foi apresentado ao PMI, que é o Instituto de Gerenciamento de Projetos, e também ao Guia do

Conhecimento em Gestão de Projetos, o PMBOK, vamos aprofundar um pouco mais os estudos e encontrar a melhor forma de como as boas práticas por eles disseminadas podem auxiliá-lo no desenvolvimento de projetos de software.

A gestão de projetos, de acordo com o PMBOK (2013), traz o gerenciamento dividido em cinco fases, também conhecidas como **ciclo de vida do projeto**:

Figura 1 – Fases do gerenciamento de projetos



Fonte: Adaptado de PMBOK (2013).





## Para saber mais

Pesquise sobre os aspectos do gerenciamento de projetos trazidos por Marconi (2007) em: MARCONI, Fábio V. **Gerenciamento de Projetos de Tecnologias de Informação**. 2. ed. Rio de Janeiro: Elsevier, 2007. Capítulo 2.


Leia também o artigo “Gestão de Projetos: a profissão do futuro”. Disponível em: <<http://www.administradores.com.br/artigos/carreira/gestao-de-projetos-a-profissao-do-futuro/75051/>>. Acesso em: 11 maio 2016.

Em iniciação, é preciso definir quem são as partes interessadas pelo projeto, ou seja, os *stakeholders* (equipe de projeto, clientes, parceiros envolvidos, funcionários da operação a **automatizar**, investidores, patrocinadores ou *sponsor*). O documento que está presente na fase de “iniciação” é o Termo de Abertura do Projeto – TAP. O Quadro 2 traz as informações que devem constar no TAP:

Quadro 2 – Componentes do Termo de Abertura de Projetos

1. Histórico de alterações de versão.	2. Justificativa do projeto.
3. Objetivos.	4. Entregáveis ou requisitos.
5. Premissas.	6. Restrições a seguir.
7. Descrição do projeto.	8. Cotas de orçamento.
9. Prévia dos riscos.	10. Marcos de entregas.
11. Stakeholders.	12. Itens para aprovação do projeto.
13. Gerente e equipe de projetos.	14. <i>Sponsor</i> .
15. Responsável pelo projeto.	

Fonte: Elaborado pelo autor.



Marconi (2007, ps. 73-74) define a importância das fases do projeto da seguinte forma. Acompanhe:



Cada fase pode ter um conjunto de subprodutos para melhor controle do gerenciamento. Em projetos de tecnologia da informação, principalmente de desenvolvimento de software, normalmente adotamos nomes como: levantamento de requisitos, análise, implementação, codificação, testes, documentação, implantação, transição, suporte, entre outros.

Ao encontro das palavras de Marconi, que muito se aproxima do que estudaremos a seguir, a análise do sistema em projetos de desenvolvimento de software, está a fase de planejamento. Planejamento é a mais complexa delas, pois envolve a elaboração do plano de gerenciamento de todas as áreas de conhecimento descritas no PMBOK (2013), a saber: integração, escopo, tempo, custos, qualidade, recursos humanos, comunicações, riscos, aquisições e partes interessadas.




Aprenda mais sobre gestão de projetos com Ricardo Vargas, autor de mais de 10 livros de gestão de projetos. Disponível em: <<http://www.ricardo-vargas.com/pt>>. Acesso em: 19 abr. 2016.

Saiba que será no escopo a definição de todas as atividades que deverão ser realizadas para que aconteça a entrega dos marcos, ou seja, dos produtos ou serviços pertinentes a uma etapa do projeto. Entendemos como escopo todos os entregáveis para que o sistema esteja completo: cada um dos módulos, cada uma de suas funcionalidades, nesta fase, serão identificados os requisitos funcionais

e não-funcionais do sistema. Lembrando que requisitos funcionais, de acordo com a Engenharia de Software, referem-se a todas as ações do sistema. Já requisitos não funcionais referem-se às especificações de ambiente, hardware e segurança da informação que deverão ser devidamente planejados, também nessa fase.

Na fase de “execução”, o foco repousa sobre: a orientação dos processos de apoio à execução do projeto; a garantia das entregas de qualidade nos prazos determinados; a mobilização de equipes, o seu treinamento e gerenciamento; por fim, o gerenciamento das comunicações, aquisições e partes interessadas.

Em “monitoramento e controle”, temos de despender maior atenção em: validar e



controlar escopo; controlar cronograma, custos, qualidade, comunicações, aquisições e partes interessada; e monitorar os riscos.

Para encerrar a apresentação das fases do gerenciamento de projetos, trazemos o foco da fase de “encerramento”, que é gerenciar e encerrar as fases, bem como encerrar as aquisições.

### Para saber mais

Sommerville (2011, p. 414) observa que as principais metas do gerenciamento de software, que reforçam a importância da Engenharia de Software e vêm ao encontro do que propomos aqui no que tange ao desenvolvimento de projetos ágeis, são: *1. Fornecer o software ao cliente no prazo estabelecido. 2. Manter os custos gerais dentro do orçamento. 3. Entregar o software que atenda às expectativas do orçamento. 4. Manter uma equipe de desenvolvimento que trabalhe bem e feliz.*

Agora que você já foi apresentado aos conceitos básicos sobre o nosso primeiro tema de estudos: “Introdução ao Gerenciamento de Projetos”, pesquise os itens sugeridos, realize as leituras indicadas e resolva os exercícios de fixação que são propostos. Bons estudos!



## Glossário

**Expertise:** conhecimento sobre algo. Experiência em um determinado segmento, produto, serviço. Domínio.

**Automatizar:** processo de transformar a atividade manual em uma auxiliada por uma máquina que pode ser um computador, um robô, ou qualquer mecanismo tecnológico e industrial que tenha sido modificado para facilitar e apoiar a atividade humana.

**Premissas:** fatores que devem ser considerados como componentes básicos, prévios, ou parte de um processo ou procedimento já existente.





# Questão para reflexão

A inserção das práticas de gerenciamento de projetos vem sendo aplicada em diversas áreas, tais como: educação, indústria, tecnologias e softwares, entre outros. Reflita sobre como é possível utilizar as premissas da gestão de projetos segundo o PMBOK e o PMI, em empresas de pequeno e médio porte que são genuinamente desenvolvedoras de produtos de software. Separe em fases e inicie o desenvolvimento do Termo de Abertura do Projeto.





## Considerações Finais

- A gestão de projetos aplicados a projetos de desenvolvimento de software deve considerar as premissas da Engenharia de Software para implementar as práticas disseminadas pelo PMI;
- projetos ágeis podem ser considerados uma das tendências em termos de dinâmica de desenvolvimento, despertando a necessidade da escolha e definição de uma metodologia de desenvolvimento ágil;
- PMI e PMBOK são, respectivamente, o Instituto de Gerenciamento de Projetos e o Guia do Conhecimento em Gerenciamento de Projetos. Existem certificações do PMI que são muito importantes a quem deseja seguir carreira em gestão de projetos e desenvolvimento de software ágil;
- o ciclo de vida de projeto deve ser considerado como um requisito de alto nível do projeto de software, ou seja, cada fase poderá gerar subprodutos, que precisam ser devidamente planejados e definidos no escopo do projeto.



## Referências

GAMA, João Neto. A nova certificação agile do PMI. Curitiba: **Mundo Project Management**. p. 54- 55, jul. 2012.

MARCONI, Fábio V. **Gerenciamento de Projetos de Tecnologia da Informação**. 2. ed. Rio de Janeiro: Elsevier, 2007.

PROJECT Management Institute. **Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK)**. 5ª ed.. São Paulo: Saraiva, 2013.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.



# Assista a suas aulas



## Aula 1 - Tema: Introdução ao Gerenciamento de Projetos - Bloco I

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/2ac0b6840096f3b5d7e754b8ed77ed89>>.



## Aula 1 - Tema: Introdução ao Gerenciamento de Projetos - Bloco II

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/834d23faf5d98a83300b1afd9c20f159>>.



# Questão 1

**1. Analise: “Alguns dos envolvidos no projeto são: o dono (sponsor) ou patrocinador, o cliente, o gerente de projeto, a empresa ou organização executora e os membros do projeto” (MARCONI, 2007, p. 74). Assinale a alternativa que contém a palavra que define os envolvidos no projeto:**

- a) Organização
- b) PMI
- c) PMBOK
- d) Stakeholders
- e) Premissas



## Questão 2

### 2. Analise as afirmações:

I – Definição de todas as atividades que deverão ser realizadas para que aconteça a entrega dos marcos, ou seja, dos produtos ou serviços pertinentes a uma etapa do projeto.

II – De acordo com a Engenharia de Software, refere-se a todas as ações do sistema.

III – Referem-se às especificações de ambiente, hardware e segurança da informação que deverão ser devidamente planejados também nessa fase.

Assinale a alternativa que apresenta respectivamente, o conceito que se associa a cada afirmação:

- a) Projetos, escopo e requisitos.
- b) Requisitos, escopo e *sponsor*.
- c) Cliente, marcos e entregas.
- d) Escopo, qualidade e prazo.
- e) Escopo, requisitos funcionais e requisitos não funcionais.





## Questão 3

### 3. Os itens a seguir são componentes básicos de:

*1. Fornecer o software ao cliente no prazo estabelecido. 2. Manter os custos gerais dentro do orçamento. 3. Entregar o software que atenda às expectativas do orçamento. 4. Manter uma equipe de desenvolvimento que trabalhe bem e feliz.*

- a) Gestão de equipes.
- b) Gestão de projetos de software.
- c) Gestão de escopo.
- d) Gestão de planejamento.
- e) Gestão de projetos.



## Questão 4

### 4. Sobre o PMI, é correto o que se afirma apenas em:

I – O Instituto de Gerenciamento de Projetos, como já mencionado anteriormente, é uma instituição sem fins lucrativos.

II – PMI oferece certificações que comprovam que essas pessoas aprenderam e aplicam as boas práticas da área.

III – Desde 1984, o PMI oferece aos associados um programa de certificação que se tornou bastante conhecido pelo seu padrão e o rigor das avaliações para obter o título. É voltada apenas aos profissionais que possuem o ensino superior.

- a) I.
- b) I e III.
- c) II e III.
- d) I e II.
- e) I, II e III.



## Questão 5

**5. Assinale a alternativa que contém o documento que deve ser entregue na primeira fase do ciclo de vida de projetos, a iniciação:**

- a) RFP (Requisição Formal de Proposta).
- b) TAP (Termo de Abertura do Projeto).
- c) EAP (Estrutura Analítica do Projeto).
- d) Plano de Gerenciamento do Projeto.
- e) Orçamento do Projeto.



# Gabarito

## 1. Resposta: D.

Comentário: stakeholders é a palavra usada pelo PMBOK para identificar as partes e pessoas interessadas no desenvolvimento do projeto.

## 2. Resposta: E.

Comentário: a primeira afirmação é referente à função do escopo diante da fase de planejamento, a segunda afirmação refere-se à definição do que são os requisitos funcionais, e a terceira afirmação refere-se efetivamente aos requisitos não funcionais que precisam ser levados em consideração na fase de planejamento do projeto de software que envolve a fase de levantamento e análise de requisitos.

## 3. Resposta: B.

Comentário: a gestão de projetos de software, seguindo os pressupostos da Engenharia de Software, estabelece as seguintes metas para o gerenciamento desses projetos:

*1. Fornecer o software ao cliente no prazo estabelecido. 2. Manter os custos gerais dentro do orçamento. 3. Entregar o software que atenda às expectativas do orçamento. 4. Manter uma equipe de desenvolvimento que trabalhe bem e feliz.*



## 4. Resposta: B.

Comentário: apenas as afirmações I e II são verdadeiras e, portanto, corretas. A afirmação III é falsa, pois as certificações podem ser retiradas para profissionais com o ensino médio completo ou ensino superior.

## 5. Resposta: B.

Comentário: o documento que está presente na fase de “iniciação” é o Termo de Abertura do Projeto – TAP. O Quadro 2 traz as informações que devem constar no TAP:

Quadro 2 – Componentes do Termo de Abertura de Projetos

1. Histórico de alterações de versão.	2. Justificativa do projeto.
3. Objetivos.	4. Entregáveis ou requisitos.
5. Premissas.	6. Restrições a seguir.
7. Descrição do projeto.	8. Cotas de orçamento.
9. Prévia dos riscos.	10. Marcos de entregas.



# Gabarito

11. Stakeholders.	12. Itens para aprovação do projeto.
13. Gerente e equipe de projetos.	14. <i>Sponsor.</i>
15. Responsável pelo projeto.	

Fonte: Elaborado pelo autor.





# Unidade 2

## Planejamento de Projetos

---

### Objetivos

Com esta aula, temos os objetivos de:

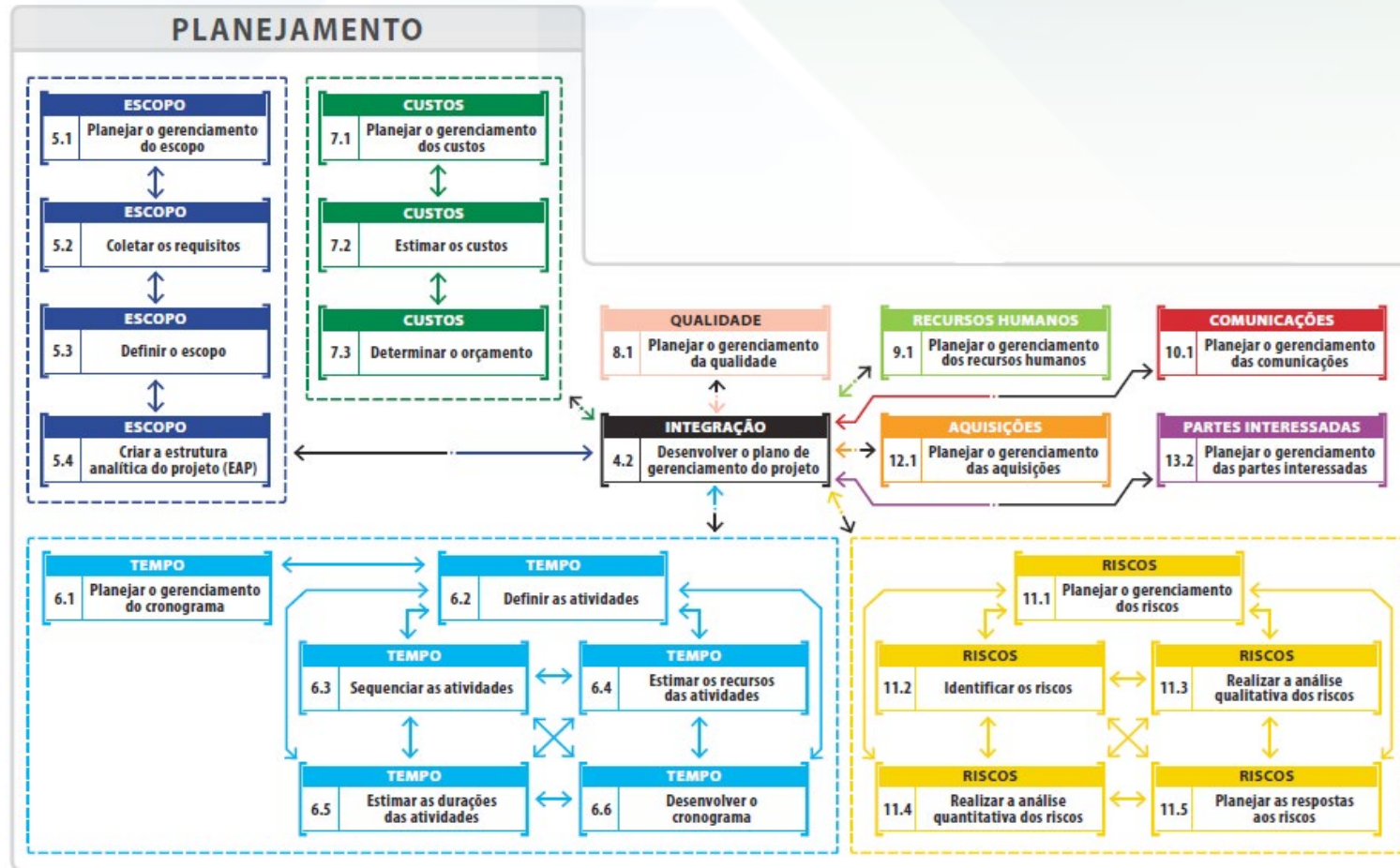
1. Apresentar todas as etapas do planejamento de projetos, de acordo com o PMBOK.
2. Incentivar a prática do planejamento do projeto associado à análise de sistemas.
3. Analisar como o planejamento do projeto pode intervir no desenvolvimento e entrega de projetos de sistemas.




## Introdução

Quando estudamos o planejamento de projetos, seguindo as premissas do guia PMBOK (2013), temos de considerar as seguintes áreas: integração, escopo, tempo, custos, qualidade, RH, comunicações, riscos, aquisições e partes interessadas. Observe na Figura 2, como é possível correlacionar as áreas de conhecimento, de acordo com Vargas:

Figura 2 – Áreas do conhecimento no planejamento do projeto



Fonte: VARGAS, Ricardo, 2014.



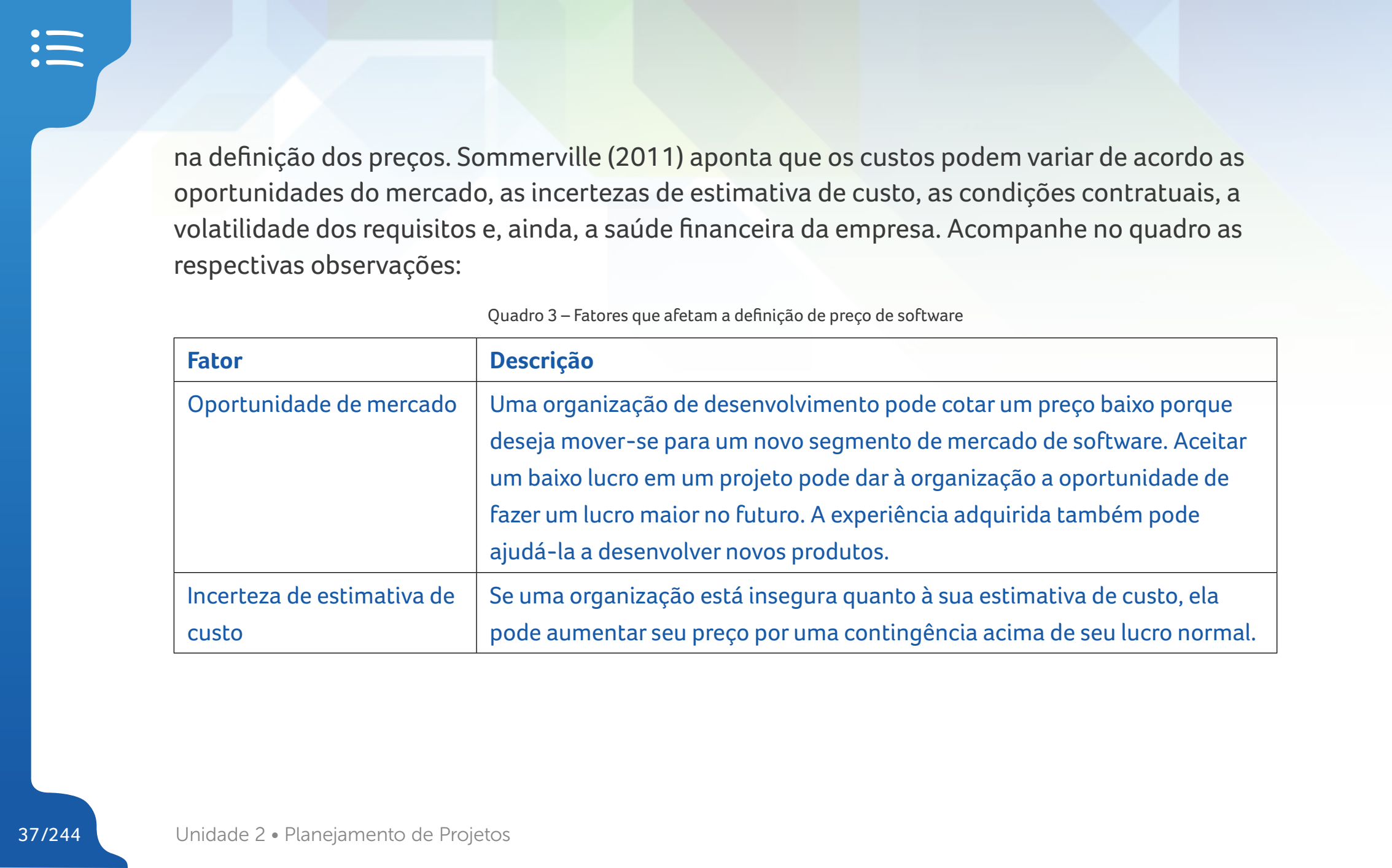
Você pode notar que a primeira análise está em mitigar o **escopo** do projeto, ou seja, passar pelas etapas de coleta e levantamento de requisitos, definição do escopo, ou seja, dos entregáveis e as respectivas tarefas para o desenvolvimento de cada um deles. A partir daí, é possível elaborar a Estrutura Analítica do Projeto (EAP) que contém todos os marcos de entregas.



O link apresenta um artigo com uma breve abordagem sobre o que é escopo e como pode ser trabalhado em projetos de desenvolvimento de software. Disponível em: <<http://www.devmedia.com.br/pmbok-gerenciamento-de-escopo-em-projetos/29787>>. Acesso em: 02 maio 2016.

A Figura 2 também mostra a necessidade de se planejar os custos do projeto, uma vez que já foram definidas as entregas, estimando os custos e fechando o orçamento.


Em projetos de desenvolvimento de software, alguns fatores podem influenciar



na definição dos preços. Sommerville (2011) aponta que os custos podem variar de acordo as oportunidades do mercado, as incertezas de estimativa de custo, as condições contratuais, a volatilidade dos requisitos e, ainda, a saúde financeira da empresa. Acompanhe no quadro as respectivas observações:

Quadro 3 – Fatores que afetam a definição de preço de software

Fator	Descrição
Oportunidade de mercado	Uma organização de desenvolvimento pode cotar um preço baixo porque deseja mover-se para um novo segmento de mercado de software. Aceitar um baixo lucro em um projeto pode dar à organização a oportunidade de fazer um lucro maior no futuro. A experiência adquirida também pode ajudá-la a desenvolver novos produtos.
Incerteza de estimativa de custo	Se uma organização está insegura quanto à sua estimativa de custo, ela pode aumentar seu preço por uma contingência acima de seu lucro normal.



Condições contratuais	Um cliente pode estar disposto a permitir que o desenvolvedor mantenha a propriedade do código-fonte e o reutilize em outros projetos. O preço cobrado pode ser maior se o código-fonte do software for entregue ao cliente.
Volatilidade de requisitos	Se os requisitos não puderem ser alterados, uma organização pode reduzir seu preço para ganhar um contrato. Após a adjudicação, pode cobrar preços elevados por alterações nos requisitos.
Saúde financeira	Os desenvolvedores com dificuldades financeiras podem baixar seus preços para ganhar um contrato. É melhor obter lucro menor do que sair do negócio. Em tempos economicamente difíceis, o fluxo de caixa é mais importante do que o lucro.

Fonte: SOMMERVILLE, 2011, p. 433.



O link apresenta um artigo que associa as práticas do PMBOK com a definição dos custos de um projeto de software:

Disponível em: <<http://www.devmedia.com.br/pmbok-trabalhando-com-gerenciamento-de-custos/31158>>. Acesso em: 1 maio 2016.

Vale ressaltar que, nesta breve apresentação, não estão sendo detalhados os fatores considerados no momento de estimar e definir os custos de um projeto de desenvolvimento de software. Por esse motivo, a leitura do artigo que foi recomendado é muito interessante para recordar alguns pontos-chave.

Na sequência e não menos importante, o planejamento apresenta ênfase quanto à gestão do tempo. Observe que a definição do cronograma está diretamente interligada à definição de atividades, ao seu sequenciamento, ou seja, à ordem de execução de acordo com a prioridade de entrega, à determinação do tempo que leva para realizar cada uma das atividades, bem como à estimativa dos recursos necessários para fazê-las. Com isso, é possível definir o cronograma do projeto.



## Para saber mais

Conheça a ferramenta Wrike, que auxilia no gerenciamento de projetos. É uma importante visão sobre o andamento das atividades e o controle do cronograma! Saiba mais em: <<http://try.wrike.com/>>. Acesso em: 1 maio 2016.

Em planejamento, também está presente o plano do gerenciamento de riscos. Esse requer atenção especial, pois considera a identificação, a análise qualitativa dos riscos, em que há a definição do impacto ao projeto se o risco vir a ocorrer, bem como o planejamento das respostas àquele determinado risco identificado, em que há uma definição de como ele pode ser tratado. Além disso, a análise quantitativa


considera impactos em custos, recursos e prazo.

Além desses, as áreas do conhecimento como integração, qualidade, plano de gerenciamento de recursos humanos e partes interessadas, plano de gestão de comunicação e aquisição também pertencem a esta importante etapa.

## Para saber mais

Leia o artigo “Gerenciamento da Integração do Projeto” e veja algumas dicas de como pode ser realizado esse trabalho. Disponível em: <<http://escritoriodeprojetos.com.br/gerenciamento-da-integracao-do-projeto.aspx>>. Acesso em: 2 maio 2016.





Agora que apresentamos os principais elementos a considerar no planejamento da gestão de projetos, você já pode ampliar seus conhecimentos vendo como trabalhar com alguns deles. Vamos lá!

## 1. Planejamento do Projeto

Você acabou de conhecer a estrutura de planejamento de projetos, de acordo com o PMBOK (2013). Mas entendeu a importância de se pensar em todas as tarefas que devem ser executadas, antes efetivamente de sair desenvolvendo o software? As recomendações de observação de cada uma das áreas do conhecimento não são em vão. Essa tarefa, o planejamento, auxilia na identificação de atividades, dos entregáveis, no controle, na integração das áreas de conhecimento, minimização de incidência de alterações de escopo, entre outros fatores que são importantes durante a execução do projeto, ou seja, da sua análise e desenvolvimento:



Em primeiro lugar, o gerente do projeto escolhe uma metodologia de desenvolvimento de sistemas adequada às características do projeto. Com base no tamanho do sistema, são feitas estimativas de prazos.



A seguir, é criada uma lista de tarefas a serem realizadas, e essa lista forma a base para o plano de trabalho do projeto. São determinadas as necessidades de pessoal, e o gerente do projeto estabelece mecanismos para coordenar a equipe do projeto ao longo de sua elaboração. Finalmente, o gerente do projeto o monitora e refina as estimativas à medida que o trabalho se desenvolve. (DENNIS et al., 2014, p. 45)

Mas planejar um projeto de software requer a escolha de uma metodologia de desenvolvimento, que permita especificar além de questões relacionadas à gestão desse projeto, identificar exatamente quais serão as funcionalidades do software, quais são os usuários, os impactos da implantação, os riscos quanto à segurança da informação, o custo da equipe de projeto, entre outros.

Para isso, é preciso levar em consideração o ciclo de vida de desenvolvimento de sistemas. Tenha sempre em mente a seguinte estrutura:

Figura 3 – Ciclo de vida de desenvolvimento de software



Fonte: Adaptado de Dennis et al., 2014, p. 11.

A fase de planejamento tem por função responder às seguintes perguntas: por que é necessário desenvolver o projeto? Como é possível estruturá-lo?

Dennis, Wixom e Roth (2014) apresentam as recomendações do planejamento ideais ao desenvolvimento de sistemas, com as suas respectivas características técnicas e o que é esperado em resultado, ou seja, em produto de desenvolvimento de software. Observe o quadro em que são apresentadas as recomendações dos autores, com a sua associação ao processo de planejamento, previsto pelo PMBOK:

Quadro 4 – Associação de planejamento da Engenharia de Software com os processos de planejamento do PMBOK

Associação com o PMBOK	Etapa	Técnica	Produtos
Planejamento do escopo	Identificar oportunidades	Identificação do projeto	Solicitação do sistema
Planejamento do escopo: coleta de requisitos	Analisar a viabilidade	Viabilidade técnica Viabilidade econômica Viabilidade organizacional	Estudo de viabilidade
Planejamento do escopo da EAP (Estrutura Analítica do Projeto)  E plano de gerenciamento de tempo	Desenvolver o plano de trabalho	Estimativa de tempo Identificação das tarefas Estrutura da divisão de trabalho Gráfico de PERT Gráfico de Gantt Gerenciamento do escopo	Plano do projeto e plano de trabalho

Associação com o PMBOK	Etapa	Técnica	Produtos
Plano de gerenciamento de recursos humanos	Compor o quadro de pessoal do projeto	Seleção de equipe Diagrama do projeto	Plano de pessoal
Plano de gerenciamento da integração e riscos	Controlar e dirigir o projeto	Repositório CASE Definição de padrões Documentação Avaliação de riscos	Lista de padrões  Avaliação de riscos

Fonte: Adaptado de Dennis et al., 2014, p. 12.

O quadro buscou associar os principais elementos que a Engenharia de Software considera na etapa de planejamento, de acordo com a visão do ciclo de vida de desenvolvimento de produto de software, com a fase de planejamento recomendada pelo PMBOK.



## Para saber mais

Leitura dos capítulos 1 e 2 do livro:

DENNIS, Alan et al. **Análise e Projeto de Sistemas**. 5. ed.. Rio de Janeiro: LTC, 2014.

O que precisa ficar claro é que cada um dos elementos mencionados em gestão de projetos baseados nas boas práticas do PMBOK tem a sua análise em seu respectivo elemento de planejamento, por exemplo, em planejamento do escopo, pode ser inserido o item de identificação do projeto.


Outro exemplo que pode ser associado é a análise de tempo de duração de atividades, com a técnica de PERT, que

está diretamente associada à elaboração do plano de gerenciamento do tempo, de acordo com o PMBOK (2013).



Compreenda de que forma a técnica PERT (Program Evaluation and Review Technique) pode auxiliar na estimativa de duração de realização de atividades em um projeto, de acordo com cenários otimista, mais provável e pessimista. Artigo disponível em: <<http://www.revistagep.org/ojs/index.php/gep/article/view/25/271>>. Acesso em: 2 maio 2016.

Do mesmo modo, quando no quadro lemos análise de viabilidade e associamos essa etapa ao processo de coleta de requisitos



do plano de gerenciamento de escopo, estamos afirmando que, para que se obtenha a análise de viabilidade técnica, econômica e organizacional, é necessário compreender as solicitações do cliente, o que ele espera do software, como esse pode auxiliar a empresa a atingir aos seus objetivos, e, ainda, se é viável de fato ter estes custos no orçamento.

Outros elementos importantes na fase de planejamento de projeto de desenvolvimento de software devem ser considerados, como, por exemplo, a definição de duração das atividades, se será dimensionada em dias ou em meses; a estimativa de recursos que serão necessários à execução de todas

as atividades relacionadas ao projeto, ou seja, quantas pessoas e quais são esses profissionais; além da identificação, através de técnicas como PERT, para determinar a duração das atividades e do projeto e, conseqüentemente, os custos de cada uma delas.

Com esses elementos, é possível identificar o **deadline** existente para a conclusão de cada atividade:



Ao planejar um projeto, você também deve definir *milestones*, ou seja, cada estágio do projeto em que pode ser feita uma avaliação de progresso. Cada *milestone* deve ser documentado por um relatório curto do qual conste o resumo dos progressos e dos trabalhos realizados. Os *milestones* podem ser associados com uma única tarefa, ou com grupos de atividades relacionadas (SOMMERVILLE, 2011, p. 438).

Nesse contexto, *milestones* também podem se referir aos entregáveis do projeto.

## 2. Planejamento Ágil

Ao trabalhar com o tema planejamento de projetos, algumas concepções podem ser abordadas. Vimos as premissas do PMBOK (2013) em relação às técnicas trazidas da Engenharia de Software para a etapa de planejamento (DENNIS, 2014), e agora vamos conhecer a abordagem do planejamento de software, não sob a ótica de planos, e sim com a possibilidade de inserção de incrementos durante o desenvolvimento do software. Aqui, portanto, será apresentada a abordagem de planejamento de software de acordo como que se conhece por planejamento ágil:





Os métodos ágeis de desenvolvimento de software são as abordagens interativas nas quais o software é desenvolvido e entregue aos clientes em incrementos. Ao contrário das abordagens dirigidas a planos, a funcionalidade desses incrementos não é planejada com antecedência, mas decidida durante o desenvolvimento (SOMMERVILLE, 2011, p. 440).

Os métodos de planejamento ágil buscam a sua justificativa nas mudanças de escopo que podem ser solicitadas pelo cliente, a considerar que as estratégias de negócios podem mudar e, ainda, para que o software em desenvolvimento possa atender a essas expectativas, precisa permitir a inserção de tais incrementos, de forma a abranger suas reais necessidades.

O tema seguinte traz com mais detalhes quais são os principais tipos de **métodos ágeis** utilizados no mercado. Recomendamos as leituras complementares e realização dos exercícios, para a fixação dos conceitos aqui apresentados. Bons estudos!



## Glossário

**Escopo:** inclui a definição de atividades necessárias para que seja possível realizar uma entrega. Ao conjunto de atividades entregáveis, estabelece-se um ponto principal que determina o controle dessa atividade, também chamado de marco. Escopo de projeto de software inclui as funcionalidades do sistema como sendo os marcos, e as tarefas necessárias à sua entrega, as atividades.

**Deadline:** o termo em inglês é muito comum na área de gestão de projetos e planejamento. A sua tradução é: prazo final. Refere-se ao último prazo disponível para a entrega do produto ou serviço.

**Métodos ágeis:** surgiram da necessidade de se acelerar o desenvolvimento de projetos de produtos de software. Muito utilizados para a realização de incrementos em projetos em andamento. Requer maior contato e comunicação com o cliente, o solicitante, o *sponsor* do projeto.



# Questão para reflexão

Como identificar a melhor abordagem para realizar o planejamento de um projeto de desenvolvimento de software?





## Considerações Finais

A fim de incentivar a adoção de uma das abordagens apresentadas, sejam pelas premissas do PMBOK, seja por um método de planejamento recomendado pela Engenharia de Software, ou, ainda, por um método de planejamento ágil é que esses foram elencados em seu material.

A ênfase se dá na escolha e determinação de pontos essenciais para o bom desenvolvimento de sistemas, que devem ser considerados nessa fase inicial e que podem ser aplicados com o apoio de uma das formas de planejamento abordadas.

A partir dessas orientações, é preciso conhecer e saber como aplicar o desenvolvimento de software sob uma perspectiva de desenvolvimento ágil, que será apresentado no próximo tema de estudos.



## Referências

DENNIS, Alan et al. **Análise e Projeto de Sistemas**. 5. ed. Rio de Janeiro: LTC, 2014.

PROJECT Management Institute. **Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK)**. 5. ed. São Paulo: Saraiva, 2013.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

VARGAS, Ricardo. Pmbok 5ª Edição. 47 Processos de Gerenciamento de de Projetos, 2014.  
Disponível em: <[www.ricardo-vargas.com](http://www.ricardo-vargas.com)>. Acesso em: 1 maio 2016.



# Assista a suas aulas



## Aula 2 - Tema: Planejamento de Projetos - Bloco I

Disponível em: <<http://fast.player.liquidplatform.com/pA-piv2/embed/dbd3957c747affd3be431606233e0f1d/315e-21029eb8da72491d0dddd78dee35>>.



## Aula 2 - Tema: Planejamento de Projetos - Bloco II

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/53747fd4eb425b7e7269b88b436f1b65>>.



# Questão 1

**1. Segundo o Guia PMBOK (2004, p. 103), “O gerenciamento do escopo trata principalmente da definição e controle do que é e do que não está incluído no projeto” (apud BRITO, Ilmário. A importância da gestão de escopo em projetos. Disponível em: <[http://www.techoje.com.br/site/techoje/categoria/detalhe\\_artigo/406](http://www.techoje.com.br/site/techoje/categoria/detalhe_artigo/406)>. Acesso em: 3 maio 2016).**

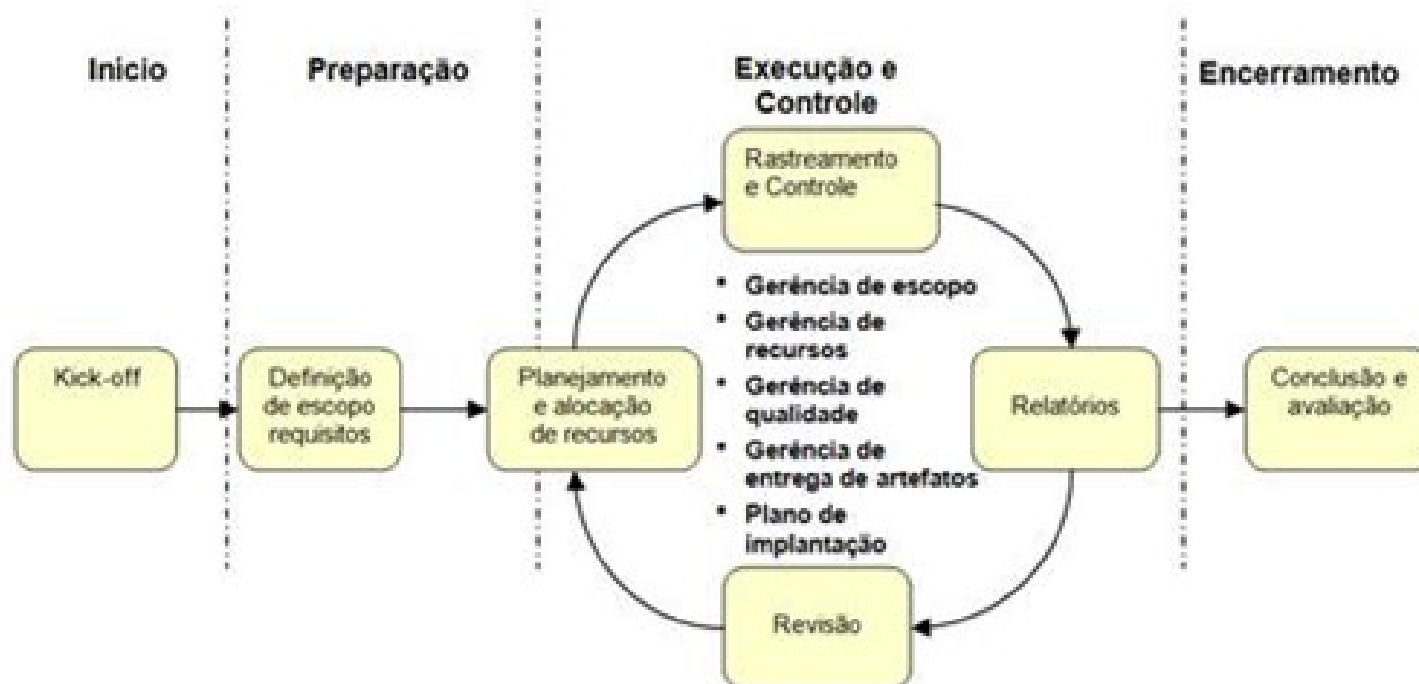
Pertence ao grupo de processos de planejamento de escopo:

- a) Coleta de requisitos; definição de escopo; criação da EAP.
- b) Planejamento de requisitos; plano de comunicação e integração.
- c) Definição de atividades; estimativa de duração e elaboração do cronograma.
- d) Plano de custos; análise PERT.
- e) Definição de equipes e EAP.

# Questão 2

## 2. Analise a figura a seguir:

Figura 4 – Análise de sistemas sob a perspectiva da gestão de projetos



Fonte: Disponível em: <<http://www.devmedia.com.br/artigo-engenharia-de-software-3-plano-de-projeto/9527>>. Acesso em: 3 maio 2016.





## Questão 2

Sobre a figura e a sua correlação com o conceito de planejamento de projetos de desenvolvimento de software, é possível considerar que:

I – O plano de projetos é parte fundamental à análise e desenvolvimento de sistemas.

PORQUE

II – O plano de projetos auxilia na integração das ações e permite o controle da evolução das atividades.

- a) A primeira afirmação está correta e a segunda não justifica a primeira.
- b) A primeira afirmação está correta e a segunda justifica a primeira.
- c) A primeira afirmação é verdadeira e a segunda é falsa.
- d) A primeira afirmação é falsa e a segunda é verdadeira.
- e) As duas afirmações estão corretas e a segunda complementa a primeira.



## Questão 3

### 3. Analise as afirmações:

I – Uma organização de desenvolvimento pode cotar um preço baixo porque deseja mover-se para um novo segmento de mercado de software.

II – O preço cobrado pode ser menor se o código-fonte do software entregue ao cliente.

III – Se os requisitos podem preço ser alterados, uma organização pode reduzir seu preço para ganhar um contrato.

Quanto ao planejamento do custo de um projeto de software, são verdadeiras as afirmações:

- a) II e III.
- b) I e II.
- c) I e III.
- d) I, II e III.
- e) Apenas III.



## Questão 4

### 4. Considere a frase abaixo:

“Ao planejar um projeto, você também deve definir milestones, ou seja, cada estágio do projeto em que pode ser feita uma avaliação de progresso. Cada milestone deve ser documentado por um relatório curto do qual conste o resumo dos progressos e dos trabalhos realizados. Os milestones podem ser associados com uma única tarefa, ou com grupos de atividades relacionadas” (SOMMERVILLE, 2011, p. 438).

Quanto aos milestones, é correto afirmar que:

- a) são identificados no plano do escopo do projeto, pois consideram os entregáveis, as atividades que devem ser realizadas para cada uma das fases;
- b) são marcos importantes ao projeto. Podem representar finalização de uma fase, transição ou conclusão de um entregável;
- c) determinam quantas pessoas precisam compor as equipes de desenvolvimento de projetos de software;
- d) podem ser substituídos pela definição do cronograma;
- e) representam integralmente os requisitos do projeto que podem ser inseridos como incrementos de atividades durante a execução do projeto.



## Questão 5

**5. Ao contrário das abordagens dirigidas a planos, a funcionalidade desses incrementos não é planejada com antecedência, mas decidida durante o desenvolvimento (SOMMERVILLE, 2011, p. 440).**

A frase acima representa uma característica de qual tipo de planejamento de projeto?

- a) Planejamento de projeto de acordo com o PMBOK 5ª ed.
- b) Planejamento do ciclo de vida do projeto.
- c) Planejamento ágil.
- d) Planejamento e análise de acordo com a Engenharia de Software.
- e) Planejamento de integração de projeto.



# Gabarito

## 1. Resposta: A.

Comentário: o grupo de processos referente à elaboração do plano de gerenciamento de escopo inclui apenas: coleta de requisitos, definição de escopo e criação da EAP (Estrutura Analítica do Projeto).

## 2. Resposta: E.

Comentário: as duas afirmações são verdadeiras e a segunda complementa a primeira, pois ambas evidenciam as competências inerentes ao plano do projeto.

## 3. Resposta: D.

Comentário: as três afirmações estão corretas. Isso se dá em função de fatores

a considerar tais como: oportunidade de mercado, incerteza de estimativa de custo e a volatilidade de requisitos.

## 4. Resposta: B.

Comentário: milestones não são atividades que precisam ser consideradas. Milestones representam os marcos e principais ocorrências do projeto, como a finalização de uma entrega, um teste de funcionalidade que represente o encerramento de uma fase de desenvolvimento, também pode ser considerado um milestone.



## 5. Resposta: C.

Comentário: os métodos de planejamento ágil buscam a sua justificativa nas mudanças de escopo que podem ser solicitadas pelo cliente, a considerar que as estratégias de negócios podem mudar e ainda que, para o software em desenvolvimento possa atender a essas expectativas, precisa permitir que a inserção de tais incrementos, de forma a atender às suas reais necessidades.



# Unidade 3

## Métodos Ágeis

---

### Objetivos

Essa tem os seguintes objetivos de aprendizagem:

1. 1. Apresentar as metodologias de desenvolvimento ágil mais comumente utilizadas no mercado de análise e desenvolvimento de sistemas.
2. 2. Fornecer subsídios para a identificação, definição e aplicação de uma metodologia de desenvolvimento ágil em análise e desenvolvimento de sistemas.
3. 3. Permitir que sejam conhecidas as boas práticas em análise de sistemas e gestão de equipes de projetos de desenvolvimento de software que adotam metodologias ágeis.



## Introdução



Para acompanhar o dinamismo das empresas em um mercado global, o investimento em softwares passou a ser uma medida que requer respostas rápidas em termos de análise, desenvolvimento e implantação de sistemas de informação.

A necessidade de uso de um software para apoio a uma determinada tarefa possibilitou que as equipes de projetos dessem maior importância à aplicação, funcionando e atendendo às solicitações e demandas, mesmo que os requisitos do sistema não estivessem totalmente identificados na realização dos incrementos, do que o projeto orientado a planos poderia oferecer. Nesse contexto é que se inserem as metodologias de desenvolvimento ágil:



Os processos de especificação, projeto e implementação são intercalados. Não há especificação detalhada do sistema, e a documentação do projeto é minimizada ou gerada automaticamente pelo ambiente de programação usado para implementar o sistema. O documento de requisitos do usuário apenas define as características mais importantes (SOMMERVILLE, 2011, p. 39).





O ritmo de desenvolvimento é mais acelerado, buscando-se realizar os incrementos das funcionalidades do sistema e disponibilizando a sua nova versão, sendo que todos os envolvidos nesse projeto, ou seja, os *stakeholders* têm uma participação mais intensiva em testes e validação do sistema, ou daquela nova funcionalidade.

Mas você pode agora se perguntar se o tempo investido e sugerido pela Engenharia de Software no planejamento e análise do sistema está em desuso. Não se trata de terem se tornado obsoletas as técnicas de análise de sistemas, como, por exemplo, o desenvolvimento em cascata, ou mesmo em espiral (vide capítulo 2 de Sommerville), e sim que apenas não

precisam ser aplicadas a todos os tipos de desenvolvimento de software. Tais técnicas que requerem maior planejamento têm sido empregadas no desenvolvimento de aplicações voltadas a governo e a empresas de grande porte com exigência de robustez em aplicações.

Para que nas empresas possa se obter uma solução em sistema em tempo recorde, que atenda a uma determinada demanda, é necessário transpor algumas barreiras e estabelecer as prioridades em termos de identificação e especificação de requisitos, bem como do próprio desenvolvimento de sistemas.

Algumas das premissas que o manifesto ágil segue incluem a dedicação ao projeto focado em pessoas e as interações que

precisam do sistema, com mais ênfase do que em processos de planejamento, análise, documentação e ferramentas de desenvolvimento. Além dessa, ainda preconizam que é mais valioso para a empresa o software em funcionamento do que a documentação abrangente. Com isso, exige-se uma participação maior do cliente durante todo o processo de desenvolvimento, de forma que o objetivo esteja mais voltado a atender às demandas de acordo com as mudanças ocorridas, do que necessariamente em planos (SOMMERVILLE, 2011).



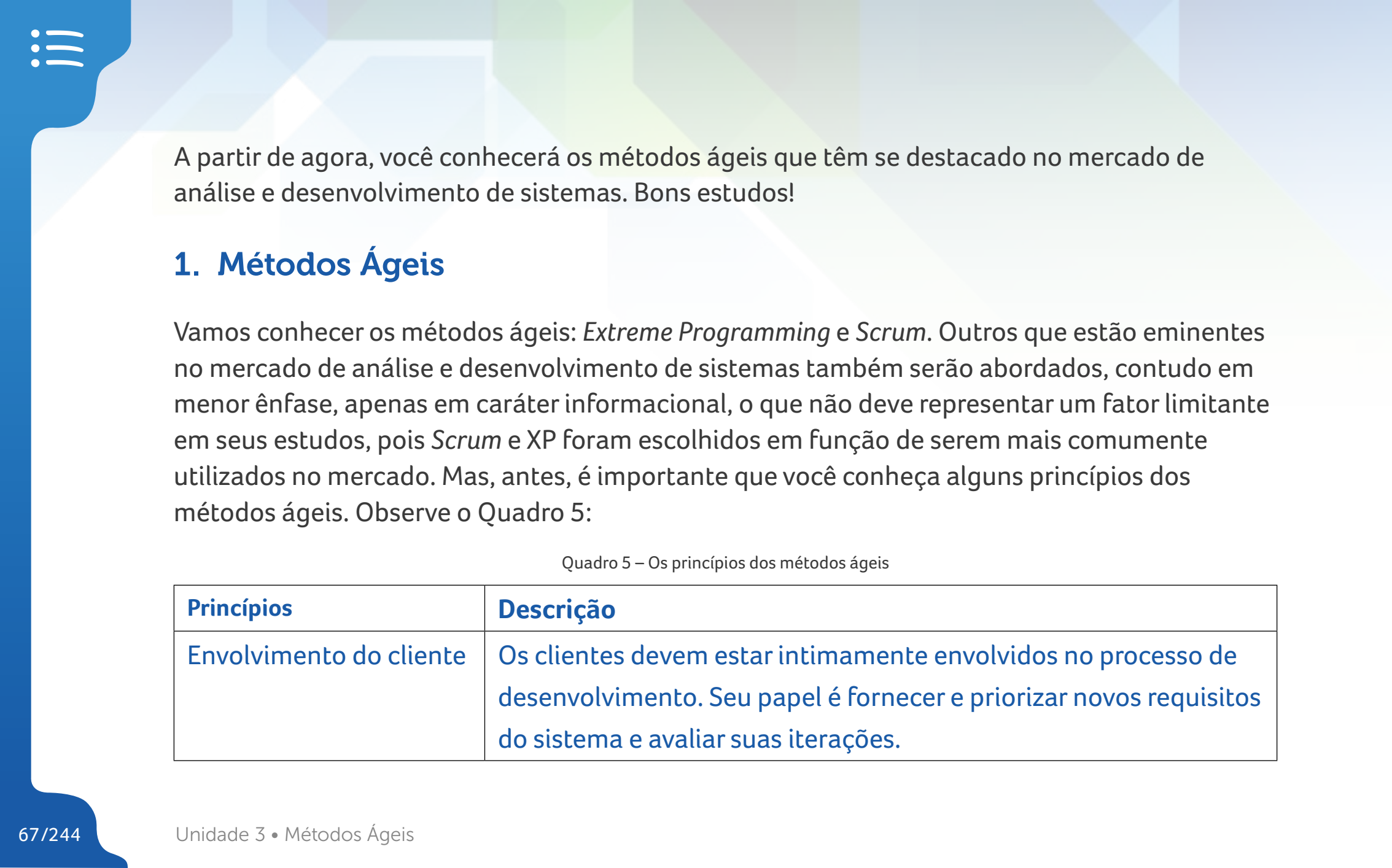
Leia sobre o manifesto ágil. Disponível em:  
<<http://www.agilemanifesto.org/iso/ptbr/>>.

Acesso em: 2 maio 2016.

## Para saber mais

Fica aqui a recomendação de leitura da edição de número 64 (Março & Abril, 2014), da revista *Mundo Ágil: Fazendo ágil funcionar em sua empresa*. Editora MAG. Disponível em: <[www.mundoj.com.br](http://www.mundoj.com.br)>. Acesso em: 2 maio 2016.

Aproveite para realizar a leitura do artigo: “Como é a implantação de métodos ágeis em sua empresa?”. Disponível em: <[http://www.infoq.com/br/news/2010/02/metodologia\\_agil\\_empresa](http://www.infoq.com/br/news/2010/02/metodologia_agil_empresa)>. Acesso em: 11 maio 2016.




A partir de agora, você conhecerá os métodos ágeis que têm se destacado no mercado de análise e desenvolvimento de sistemas. Bons estudos!

## 1. Métodos Ágeis

Vamos conhecer os métodos ágeis: *Extreme Programming* e *Scrum*. Outros que estão eminentes no mercado de análise e desenvolvimento de sistemas também serão abordados, contudo em menor ênfase, apenas em caráter informacional, o que não deve representar um fator limitante em seus estudos, pois *Scrum* e XP foram escolhidos em função de serem mais comumente utilizados no mercado. Mas, antes, é importante que você conheça alguns princípios dos métodos ágeis. Observe o Quadro 5:

Quadro 5 – Os princípios dos métodos ágeis


Princípios	Descrição
Envolvimento do cliente	Os clientes devem estar intimamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar suas iterações.



Princípios	Descrição
Entrega incremental	O software é desenvolvido em incrementos com o cliente, especificando os requisitos para serem incluídos em cada um.
Pessoas, não processos	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Membros da equipe devem desenvolver suas próprias maneiras de trabalhar, sem processos prescritivos.
Aceitar mudanças	Deve-se ter em mente que os requisitos do sistema vão mudar. Projete o sistema de maneira a acomodar essas mudanças.
Manter a simplicidade	Focalize a simplicidade, tanto do software a ser desenvolvido quanto do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.

Fonte: SOMMERVILLE, 2011, p. 40.

Como especificado, a objetividade e atenção ao projeto de análise e desenvolvimento ágil buscam, além da aceleração e entrega mais rápida do produto de software solicitado, a




integração das equipes com o cliente e vice-versa, pois acredita que tal parceria de desenvolvimento pode ajudar a minimizar erros de requisitos e, portanto, da entrega de funcionalidades do sistema.

No entanto, nem sempre o cliente está disponível à medida que se necessita de informações ou mesmo a sua validação e testes do sistema. Ou ainda, outra dificuldade desse modelo, está em encontrar o perfil de profissional que saiba trabalhar em equipe e manter um relacionamento mais estreito com os demais *stakeholders* do projeto (SOMMERVILLE, 2011).

Para o caso da administração das mudanças e das demandas que isso significa para o ambiente de sistemas e

suas respectivas funcionalidades, há um desafio nesse contexto que está em definir o que é prioridade de desenvolvimento e qual área da empresa atender primeiro. No entanto, já é sabido que os sistemas de informação existem a fim de auxiliar na tomada de decisões em empresas e, também, para que possam acompanhar as estratégias que são estabelecidas por ela, mas, para que isso aconteça, é necessário que as regras de negócios sejam mapeadas e incrementadas no sistema em forma de nova funcionalidade. Então, o alinhamento de informações entre as áreas e o atendimento aos prazos tornam-se de extrema importância nesse processo.



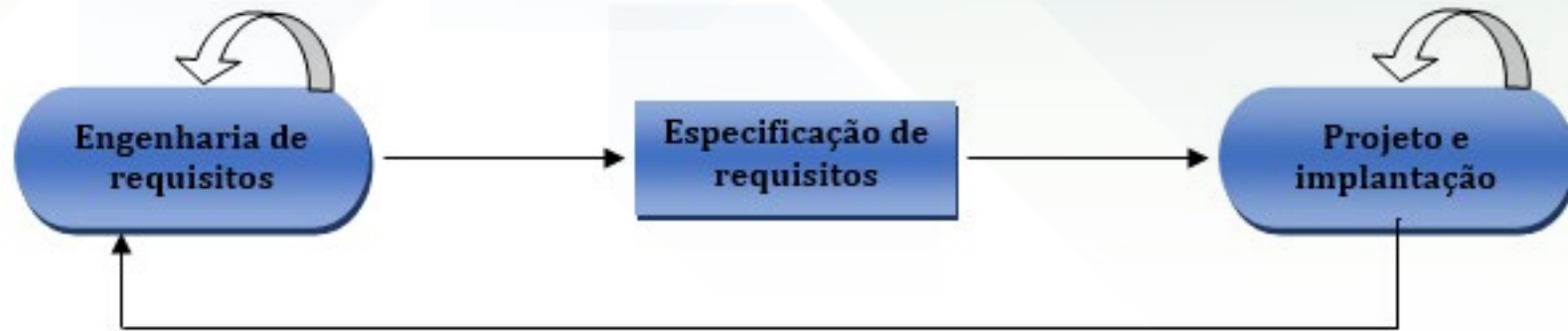
Note que, nesse sentido, é possível retomar a importância de se inserir um olhar de gestão sobre essa demanda que a área de gestão de projetos pode auxiliar mais fortemente, principalmente se conseguir inserir as premissas e boas práticas contidas no PMBOK.



Faça a leitura do artigo: “Melhores práticas para a gerência de projetos ágeis”, da revista *Engenharia de Software Magazine*, disponibilizado pela DevMedia. Disponível em: <<http://www.devmedia.com.br/melhores-praticas-para-a-gerencia-de-projetos-ageis-revista-engenharia-de-software-magazine-48/24378>>. Acesso em: 2 maio 2016.

Mas, afinal, você sabe a diferença estrutural no que tange à análise e desenvolvimento de sistemas orientada a planos do desenvolvimento ágil? Observe as figuras e analise-as:

Figura 5 – Especificação dirigida a planos



Fonte: Adaptado de Sommerville, 2011, O. 43.

Observe que, na Figura 5, há uma orientação do projetos a planos, então temos o plano de engenharia de requisitos mais detalhado, as especificações de requisitos mais documentadas, inclusive, para, depois, ocorrer a implantação. No entanto, esse rigor em documentação e análise pode demandar um tempo que não há na realidade de muitas empresas. Agora, veja a Figura 6 e o que é proposto com os métodos de desenvolvimento ágil:

Figura 6 – Especificação ágil



Fonte: Adaptado de Sommerville, 2011, O. 43.


Como é possível observar na Figura 6, temos a identificação dos requisitos e na sequência a implantação da funcionalidade no sistema. Isso faz com que a integração e comunicação entre os *stakeholders* tenham de ocorrer de forma mais aproximada e frequente.

Conheça, a seguir, os principais modelos de desenvolvimento ágil utilizados no mercado.

### 1.1 *Extreme Programming*

Essa é uma metodologia que foca, além do desenvolvimento ágil e suas premissas, a qualidade do projeto em que será aplicada, pois a análise e desenvolvimento do produto de software em questão, também conhecida como XP, contempla a valorização da objetividade e simplicidade





do projeto entregue, ou seja, que seja mais simples para o usuário e totalmente funcional em termos de arquitetura e aplicação das regras de negócios.

Alguns conceitos fortemente envolvidos nessa metodologia de desenvolvimento ágil são:


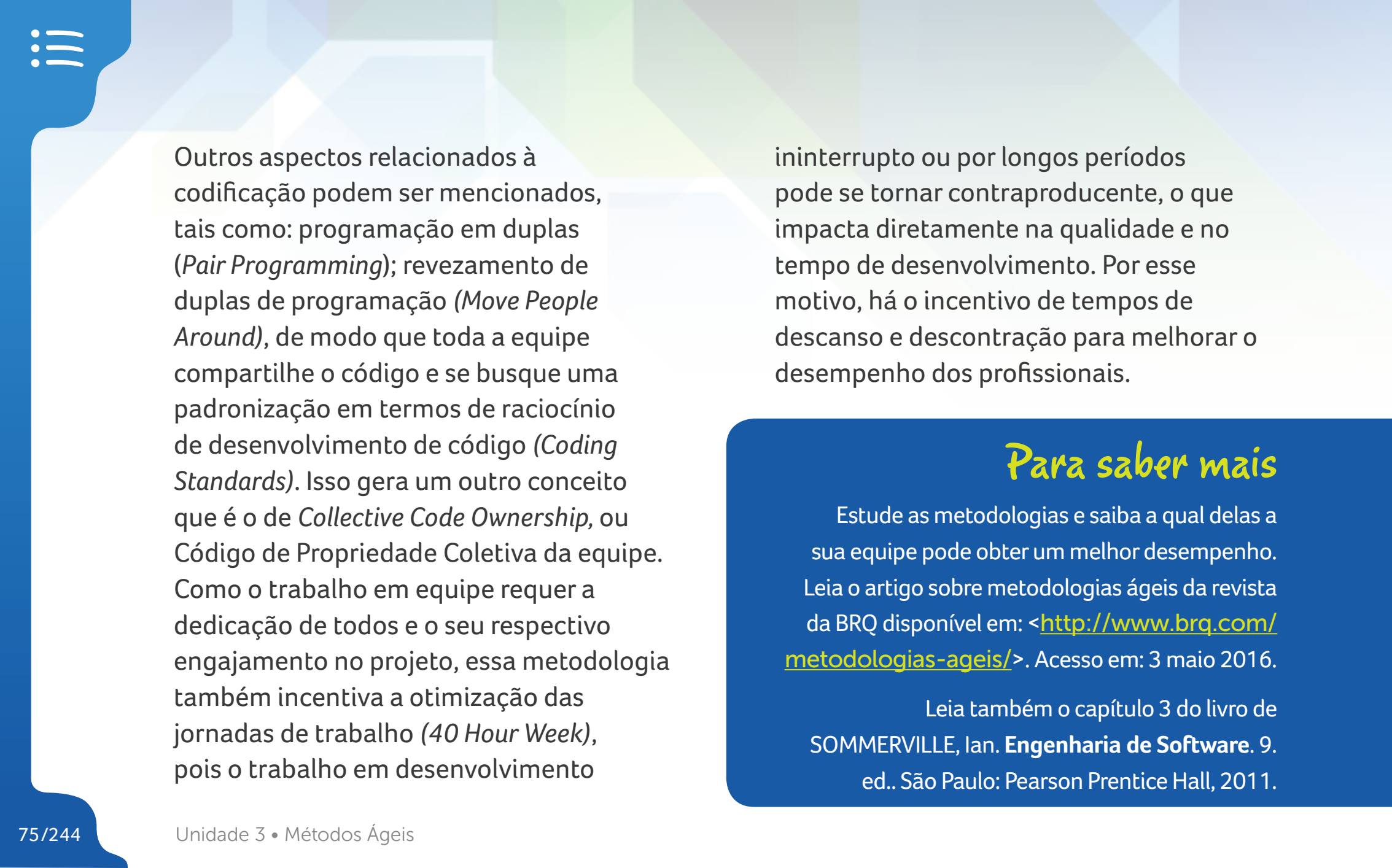
- comunicação;
- **feedback**;
- desenvolvimento por parte da equipe de trabalho de habilidades relacionadas a comportamentos e atitudes.

Quadro 6 – Boas práticas de XP

1.	O cliente precisa estar sempre disponível.
2.	Utiliza metáforas no projeto, ou seja, aproxima ao máximo a funcionalidade do incremento à área a que se aplica ou processo a que se destina apoiar.
3.	Planejamento do jogo com uso de histórias ou <i>use stories</i> , em que há uma reunião utilizando recursos para definição de textos claros da funcionalidade do incremento, em que pode ser utilizada a notação UML para a definição e especificação das regras de negócios. Isso permite, inclusive, a identificação das atividades que deverão ser realizadas para a entrega das histórias e o seu acompanhamento de perto no cronograma de desenvolvimento do projeto.

4. Utiliza-se de pequenas versões, também conhecidas como *small releases*. Essa abordagem foca no envio ao cliente das versões que são concluídas, de forma que possa validar a funcionalidade, o incremento realizado.
5. Também vale-se dos testes de aceitação (*acceptance tests*), que foca o requisito levantado inicialmente e a verificação se a versão atendeu à solicitação real do cliente.
6. *The first design* ou primeiros testes: aplica-se esse conceito para a realização de testes por funcionalidade, ou seja, unitários. Estes visam atender às regras de negócios que foram solicitadas. Dessa maneira, busca-se a redução de erros de programação.
7. Integração contínua, ou *continuous integration*, indica que a cada incremento realizado deve ser feito um teste unitário da funcionalidade, até que se obtenha a totalidade da integração.
8. Simplicidade, ou conhecida como *Simple Desing*, já mencionada como sendo interligada à programação que visa facilitar a utilização do sistema, ou seja, tornar fácil ou ao menos amigável a aplicação ao usuário.
9. Refatoração ou *refactoring*: essa é uma premissa que considera fortemente aspectos de qualidade e melhoria contínua do código do programa.

Fonte: Adaptado de Pimentel, Extreme Programming- conceitos e práticas. Disponível em: <<http://www.devmedia.com.br/extremeprogrammingconceitosepraticas/1498>>. Acesso em: 2 maio 2016.



Outros aspectos relacionados à codificação podem ser mencionados, tais como: programação em duplas (*Pair Programming*); revezamento de duplas de programação (*Move People Around*), de modo que toda a equipe compartilhe o código e se busque uma padronização em termos de raciocínio de desenvolvimento de código (*Coding Standards*). Isso gera um outro conceito que é o de *Collective Code Ownership*, ou Código de Propriedade Coletiva da equipe. Como o trabalho em equipe requer a dedicação de todos e o seu respectivo engajamento no projeto, essa metodologia também incentiva a otimização das jornadas de trabalho (*40 Hour Week*), pois o trabalho em desenvolvimento

ininterrupto ou por longos períodos pode se tornar contraproducente, o que impacta diretamente na qualidade e no tempo de desenvolvimento. Por esse motivo, há o incentivo de tempos de descanso e descontração para melhorar o desempenho dos profissionais.

## Para saber mais

Estude as metodologias e saiba a qual delas a sua equipe pode obter um melhor desempenho. Leia o artigo sobre metodologias ágeis da revista da BRQ disponível em: <<http://www.brq.com/metodologias-ageis/>>. Acesso em: 3 maio 2016.

Leia também o capítulo 3 do livro de SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed.. São Paulo: Pearson Prentice Hall, 2011.

## 1.2 Scrum

Com o objetivo de estreitar o relacionamento das equipes de trabalho com o cliente e, ainda, otimizar o tempo disponível para a entrega do incremento solicitado, com a qualidade necessária desse produto de software, a metodologia de desenvolvimento ágil contempla algumas premissas, como:

- identificação das funcionalidades do projeto (*Product Backlog*);
- elaboração de uma lista de tarefas chamada de *Sprint Backlog*;
- valorização da prática de reuniões diárias para alinhamento de informações entre as equipes e clientes;

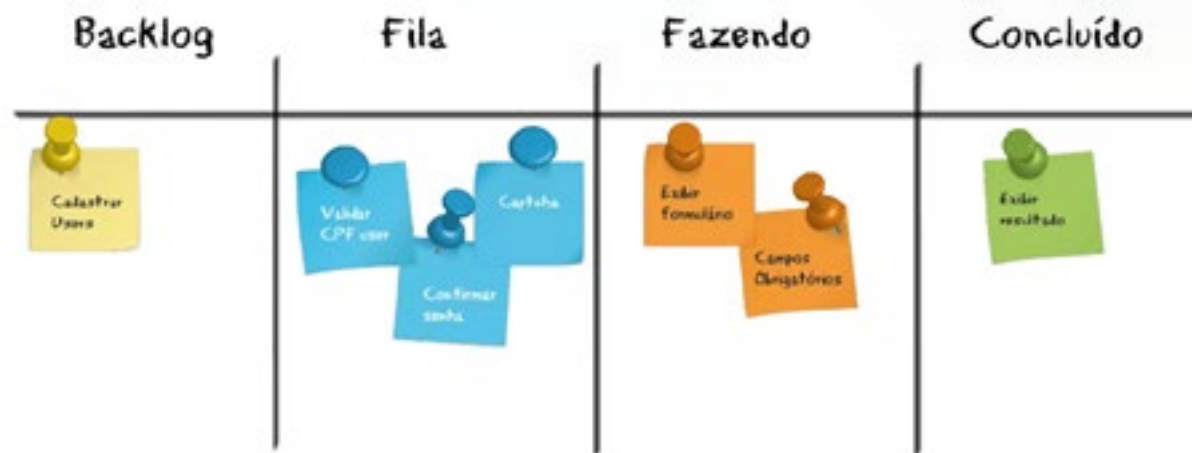
- priorização da solicitação para a sua entrega integral ou seja, produto ou funcionalidade concluída (*Potentially Shippable Product Increment*).

Quando se implementa a metodologia *Scrum*, tem-se em mente que os projetos serão divididos em ciclos mensais, normalmente, e devem, de acordo com essa metodologia, ser analisados, desenvolvidos e implementados, em um período de duas a quatro semanas (2 a 4). Esses ciclos de desenvolvimento de incrementos são chamados de *Sprint*.

Para que sejam identificadas as funcionalidades do projeto, é realizada uma reunião inicial de planejamento chamada de *Sprint Planning Meeting*, que priorizará as entregas possíveis de serem realizadas

no período de duas a quatro semanas. Podem separar as entregas em um quadro de trabalho chamado *Kanban*, que separa em uma fila as atividades: a fazer, em andamento, em teste e concluído. Observe a Figura 7, que ilustra essa informação:

1.1. Figura 7 – Kanban, quadro de tarefas




Fonte: BRQ, Metodologias ágeis. Disponível em: <<http://www.brq.com/metodologias-ageis>>. Acesso em: 2 maio 2016.

Para as atividades que não devem ultrapassar o período de 24 horas de desenvolvimento, é elaborado um gráfico que permite o seu acompanhamento, chamado de *Burn Down Chart*. Esse gráfico considera basicamente a quantidade de tarefas do *Sprint* pelos dias correspondentes ao ciclo. Veja o exemplo de gráfico *Burn down*, a seguir:

Figura 8 – Gráfico de acompanhamento de atividades no *Sprint*



Fonte: BRQ, Metodologias ágeis. Disponível em: <<http://www.brq.com/metodologias-ageis>>. Acesso em: 2 maio 2016.



Após cada *Sprint*, ou ciclo de desenvolvimento, é realizada uma outra reunião de alinhamento e encerramento da atividade, chamada de *Sprint Review Meeting*. Nessa reunião, podem ser consideradas inclusive as lições aprendidas com a implantação da funcionalidade.

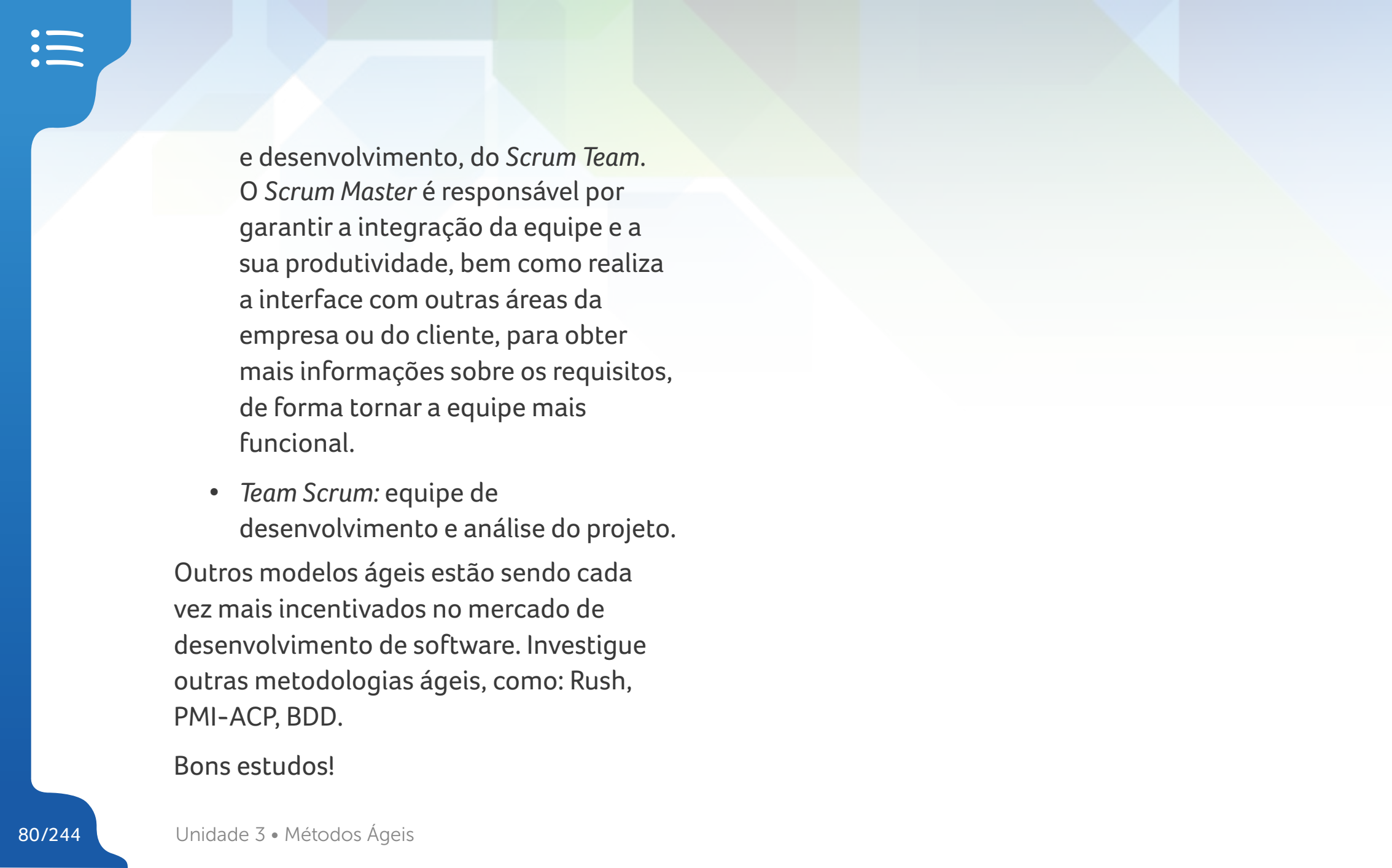


O artigo mostra os principais aspectos relacionados à metodologia de desenvolvimento ágil *Scrum*. Disponível em: <[https://endeavor.org.br/scrum/?esvt=-b&esvq=\\_cat%3Aendeavor.org.br&esvadt=999999---1&esvcrea=75514462525&esvplace=&esvd=c&esvaid=50078&gclid=COe5gPDS0swCFQGAKQodmRgLvg](https://endeavor.org.br/scrum/?esvt=-b&esvq=_cat%3Aendeavor.org.br&esvadt=999999---1&esvcrea=75514462525&esvplace=&esvd=c&esvaid=50078&gclid=COe5gPDS0swCFQGAKQodmRgLvg)>. Acesso em: 11 maio 2016.

Além da metodologia *Scrum* trazer especificadas as atividades, os períodos de trabalho em ciclos, também considera três funções principais quanto à estrutura da equipe. São eles:

- *Product Owner*, que atua no projeto diretamente com o acompanhamento do retorno financeiro do produto (**ROI**) realizando as estimativas necessárias, a priorização das tarefas de acordo com o tempo disponível e tem autonomia para alterar os requisitos de um ciclo (*Sprint*), bem como aceitar ou não o resultado da equipe. Seria o papel de um gerente de projetos.
- *Scrum Master*, que atua mais fortemente com a equipe de análise





e desenvolvimento, do *Scrum Team*. O *Scrum Master* é responsável por garantir a integração da equipe e a sua produtividade, bem como realiza a interface com outras áreas da empresa ou do cliente, para obter mais informações sobre os requisitos, de forma tornar a equipe mais funcional.

- *Team Scrum*: equipe de desenvolvimento e análise do projeto.

Outros modelos ágeis estão sendo cada vez mais incentivados no mercado de desenvolvimento de software. Investigue outras metodologias ágeis, como: Rush, PMI-ACP, BDD.

Bons estudos!





## Glossário

**ROI (Return on Invest):** retorno sobre investimento. Essa é uma forma de verificar o impacto do projeto, como está a sua saúde financeira perante a empresa, se está positiva ou negativa.

**Feedback:** utiliza-se esse termo do inglês para orientar uma necessidade de retorno de informação. Pode estar relacionada a comportamento, processo, ou mesmo gestão de equipes.

**Estórias:** termo utilizado na metodologia *Scrum*, para descrever, narrar, a ação do incremento que deverá ser inserido no sistema.



# Questão para reflexão

Em empresas, tem-se tornado cada vez mais frequente e necessário adotar uma metodologia de desenvolvimento ágil, seja em função de inserção de novas funcionalidades em tempo hábil, seja para criar maior integração entre as equipes de análise e desenvolvimento de sistemas. Diante desse contexto e das metodologias ágeis disponíveis no mercado, pesquise sobre XP, *Scrum*, PMI-ACP, BDD (*Behavior Driven Development*) ou Desenvolvimento Orientado por Comportamento. Dentre elas, qual considera realmente factível em cenários de mudanças contínuas em termos de regras de negócios e atualização de sistemas? Que estratégia adotaria para minimizar a incidência de incrementos no sistema?





# Questão para reflexão

Leitura recomendada:

Artigo 1: “Scrum e BDD: o casamento perfeito”. Disponível em: <<http://www.devmedia.com.br/scrum-e-bdd-o-casamento-perfeito/28174>>. Acesso em: 4 maio 2016.

Artigo 2: “Rush: método ágil de gerenciamento de projetos priorizando prazo”. Disponível em: <<http://pmpath.com.br/rush-metodo-agil-de-gerenciamento-de-projetos-priorizando-prazo/>>. Acesso em: 4 maio 2016.





## Considerações Finais

Prezado aluno, neste material, pudemos conhecer algumas das metodologias ágeis mais utilizadas no mercado de análise e desenvolvimento de sistemas. Também fomos apresentados a outras tecnologias que estão emergindo nesse cenário como PMI-ACP, RUSH e BDD.

Com isso, espera-se que consiga identificar a melhor metodologia, de acordo com o ambiente organizacional, que se adéque às necessidades de desenvolvimento de produtos de software que forem demandadas.

Dessa forma, é preciso avançar os estudos e conhecer UML (Unified Modeling Language), tema de nossa próxima aula. Então, até a próxima e bons estudos!



## Referências

BRQ. Metodologias ágeis. Disponível em: <<http://www.brq.com/metodologiasag>>. Acesso em: 2 maio 2016.

PIMENTEL, Manoel. DevMedia. Extreme Programming – conceitos e práticas. Disponível em: <<http://www.devmedia.com.br/extremeprogrammingconceitosepraticas/1498>>. Acesso em: 2 maio 2016.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.



# Assista a suas aulas



## Aula 3 - Tema: Métodos Ágeis - Bloco I

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f-1d/692e9167da5cc4ce84cdad9c97544013>>.



## Aula 3 - Tema: Métodos Ágeis - Bloco II

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f-1d/0bfd22fbd720128854a0e675eb13c284>>.



# Questão 1

## 1. Analise a frase e assinale a alternativa que contém exatamente, os conceitos que podem estar associados ao desenvolvimento ágil:

“Os processos de especificação, projeto e implementação são intercalados. Não há especificação detalhada do sistema, e a documentação do projeto é minimizada ou gerada automaticamente pelo ambiente de programação usado para implementar o sistema. O documento de requisitos do usuário apenas define as características mais importantes” (SOMMERVILLE, 2011, p. 39).

- a) A documentação quando se utiliza uma metodologia ágil é realizada em detalhes, devido à importância da etapa de levantamento de requisitos e planejamento das ações do sistema.
- b) Em metodologia orientada a planos é tão comum quanto na metodologia ágil que o levantamento de requisitos seja, com frequência, identificado e implementado, não requerendo maior dispêndio de tempo em identificar requisitos mais específicos, o que geralmente não ocorre nas aplicações.
- c) Algumas das premissas que o manifesto ágil segue incluem a dedicação ao projeto focado em pessoas e as interações que precisam do sistema, com mais ênfase do que em processos de planejamento, análise, documentação e ferramentas de desenvolvimento.



# Questão 1

- d) Sempre há a necessidade de descrever de acordo com a metodologia ágil, uma comparação com outro modelo de desenvolvimento, como, por exemplo, em cascata, conforme recomenda a Engenharia de Software.
- e) Não é necessário ter preocupações com documentação de incrementos que são inseridos nos sistemas, pois a metodologia ágil desconsidera a sua importância.





## Questão 2

### 2. Analise as afirmações e assinale a alternativa correta:

I – Dedicção ao projeto focado em pessoas e as interações que precisam do sistema, com mais ênfase do que em processos de planejamento, análise, documentação e ferramentas de desenvolvimento.

II – Em métodos ágeis, é mais valioso para a empresa o software em funcionamento do que a documentação abrangente.

III – Requer uma participação maior do cliente durante o processo de desenvolvimento, de forma que o objetivo esteja mais voltado a atender às demandas de acordo com as mudanças ocorridas do que necessariamente a planos.

São verdadeiras apenas:

- a) I.
- b) I e II.
- c) II e III.
- d) I, II e III.
- e) I e III.



## Questão 3

**3. Analise as informações a seguir e assinale a alternativa que contém a metodologia ágil que elas representam:**

- Identificação das funcionalidades do projeto (*Product Backlog*);
  - elaboração de uma lista de tarefas chamada de *Sprint Backlog*;
  - valoriza a prática de reuniões diárias para alinhamento de informações entre as equipes e clientes;
  - a priorização da solicitação para a sua entrega integral ou seja, produto ou funcionalidade concluída (*Potentially Shippable Product Increment*).
- a) *Extreme Programming*
  - b) Modelo em cascata
  - c) *Scrum*
  - d) UML
  - e) PMI-ACP



## Questão 4

**4. As frases abaixo representam boas práticas em XP. Identifique as palavras que completam sem prejuízos as lacunas das frases, considerando o seu real significado para a metodologia em questão:**

I – Utiliza \_\_\_\_\_ no projeto, ou seja, aproxima ao máximo a funcionalidade do incremento à área a que se aplica ou processo a que se destina apoiar.

II – Planejamento do jogo com uso de \_\_\_\_\_ em que há uma reunião utilizando recursos para definição de textos claros da funcionalidade do incremento, em que pode ser utilizada a notação UML para a definição e especificação das regras de negócios.

III – \_\_\_\_\_ é uma premissa que considera fortemente aspectos de qualidade e melhoria do código do programa.

- a) releases / histórias / refatoração
- b) simplicidade / histórias / metáforas
- c) Product Backlog / histórias / metáforas
- d) metáforas / histórias / refatoração
- e) Scrum Master / Product Owner / Sprint



## Questão 5

### 5. Associe os elementos referentes à metodologia ágil Scrum no quadro:

<i>I – Sprint</i>	( ) Esse gráfico considera basicamente a quantidade de tarefas do <i>Sprint</i> pelos dias correspondentes ao ciclo.
<i>II – Burn Down Chart</i>	( ) Identificação das funcionalidades do projeto.
<i>III – Sprint Review Meeting</i>	( ) Ciclo de desenvolvimento
<i>IV – Product Backlog</i>	( ) Reunião de alinhamento e encerramento da atividade

A sequência correspondente a cada conceito de Scrum é representada no quadro da seguinte forma:

- a) IV, II, I e I.
- b) III, IV, I e II.
- c) II, IV, I e III.
- d) I, II, III e IV.
- e) IV, II, I e III.



# Gabarito

## 1. Resposta: C.

Comentário: apenas a alternativa C é a correta, pois apresenta uma das premissas básicas que os métodos ágeis de desenvolvimento consideram, tais como: foco em pessoas e as interações que precisam do sistema, com mais ênfase do que em processos de planejamento.

## 2. Resposta: D.

Comentário: todas as afirmações são verdadeiras.

Em métodos ágeis, é mais valioso para a empresa o software em funcionamento do que a documentação abrangente. Requer uma participação maior do cliente

durante o processo de desenvolvimento, de forma que o objetivo esteja mais voltado a atender às demandas de acordo com as mudanças ocorridas do que necessariamente em planos.

## 3. Resposta: B.

Comentário: as afirmações são características da metodologia ágil *Scrum*.

- Identificação das funcionalidades do projeto (*Product Backlog*);
- elaboração de uma lista de tarefas chamada de *Sprint Backlog*;
- valorização da prática de reuniões diárias para alinhamento de informações entre as equipes e clientes;



- priorização da solicitação para a sua entrega integral ou seja, produto ou funcionalidade concluída (*Potentially Shippable Product Increment*).

## 4. Resposta: D.

Comentário: veja abaixo as características apresentadas de acordo com as respectivas palavras que preenchem as lacunas das afirmações:

- I. Utiliza **metáforas** no projeto, ou seja, aproxima ao máximo a funcionalidade do incremento à área a que se aplica ou processo a que se destina apoiar.

II. Planejamento do jogo com uso de **estórias** em que há uma reunião utilizando recursos para definição de textos claros da funcionalidade do incremento, em que pode ser utilizada a notação UML para a definição e especificação das regras de negócios.

III. **Refatoração** é uma premissa que considera fortemente aspectos de qualidade e melhoria do código do programa.



## 5. Resposta: C.

Comentário: veja abaixo o quadro completo:

<i>I – Sprint</i>	( II ) Esse gráfico considera basicamente a quantidade de tarefas do <i>Sprint</i> pelos dias correspondentes ao ciclo.
<i>II – Burn Down Chart</i>	( IV ) Identificação das funcionalidades do projeto.
<i>III – Sprint Review Meeting</i>	( I ) Ciclo de desenvolvimento
<i>IV – Product Backlog</i>	( III ) Reunião de alinhamento e encerramento da atividade



# Unidade 4

## Paradigma Orientado a Objetos: Histórico, Características e Conceitos

---

### Objetivos

Com essa aula, temos os objetivos de:

1. Apresentar, conceituar e contextualizar o paradigma da orientação a objetos.
2. Incentivar o aprendizado contínuo e aplicado para a análise de sistemas sob a perspectiva de projetos ágeis.
3. Abordar os principais aspectos da orientação a objetos de forma que o aluno possa conhecer e aprender a utilizar as ferramentas de análise de sistemas.





## Introdução


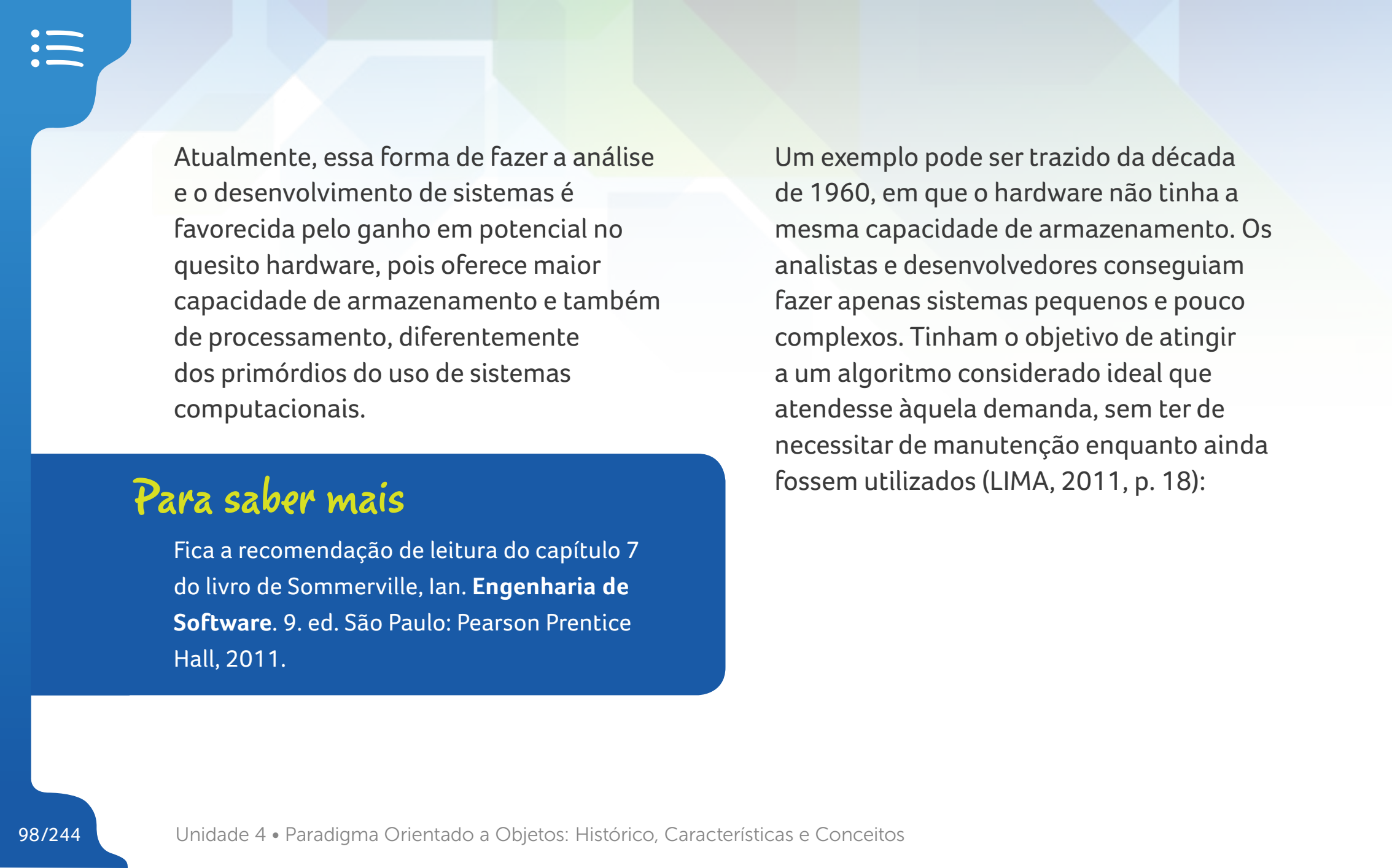
A cada dia, as empresas se veem com a necessidade de gerenciar as informações de seus negócios e obter, através do apoio dos sistemas de informação, respostas rápidas para as tomadas de decisão e estratégias que precisam adotar.

Nesse sentido, o desenvolvimento de sistemas precisa atender a tais expectativas, porém, para que isso aconteça, os sistemas devem, além de planejados em um nível de projeto com data para iniciar e para encerrar, ser analisados sob uma visão micro que considere cada uma das funcionalidades desse sistema, ou seja, que seja capaz de demonstrar em sua fase de projeto, e de forma clara, o que deverá ser entregue em termos de produto de software.

Com o intuito de apresentar uma forma de organizar os requisitos do sistema, vamos conhecer como o paradigma da orientação a objetos pode contribuir para estabelecer essa visão da estrutura do sistema, como as funcionalidades e usuários interagem, e como será essa interação, seus impactos e sequência lógica:



A abordagem orientada a objeto considera um sistema como uma coleção de objetos autocontidos, que inclui tanto dados como processos. As metodologias tradicionais de análise e o design de sistemas são centrados em dados ou em processos (DENNIS et al., 2014, p. 490).



Atualmente, essa forma de fazer a análise e o desenvolvimento de sistemas é favorecida pelo ganho em potencial no quesito hardware, pois oferece maior capacidade de armazenamento e também de processamento, diferentemente dos primórdios do uso de sistemas computacionais.

## Para saber mais

Fica a recomendação de leitura do capítulo 7 do livro de Sommerville, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

Um exemplo pode ser trazido da década de 1960, em que o hardware não tinha a mesma capacidade de armazenamento. Os analistas e desenvolvedores conseguiam fazer apenas sistemas pequenos e pouco complexos. Tinham o objetivo de atingir a um algoritmo considerado ideal que atendesse àquela demanda, sem ter de necessitar de manutenção enquanto ainda fossem utilizados (LIMA, 2011, p. 18):




Já na década de 60, começaram a surgir linguagens como *Simula* e *Smaltalk*, que permitiam sustentar o princípio de que os programas deviam ser estruturados com base no problema a ser resolvido, fundamentando a construção nos objetos de negócio em si.

Como evolução desse modelo de desenvolvimento de sistemas, a década de 1970 trouxe o conceito de programação modular, em que o uso de funções era o ponto-chave para o programa. O foco, então, estava em estabelecer procedimentos de execução das ações do programa, e não na manipulação e administração dos dados (LIMA, 2011):



A análise e o desenvolvimento fundamentavam-se no computador e na linguagem de programação, isto é, na solução e não no problema a ser resolvido – o ponto central eram os procedimentos e as funções, implementados em blocos estruturados, com comunicação por passagem de dados (LIMA, 2011, p. 17).



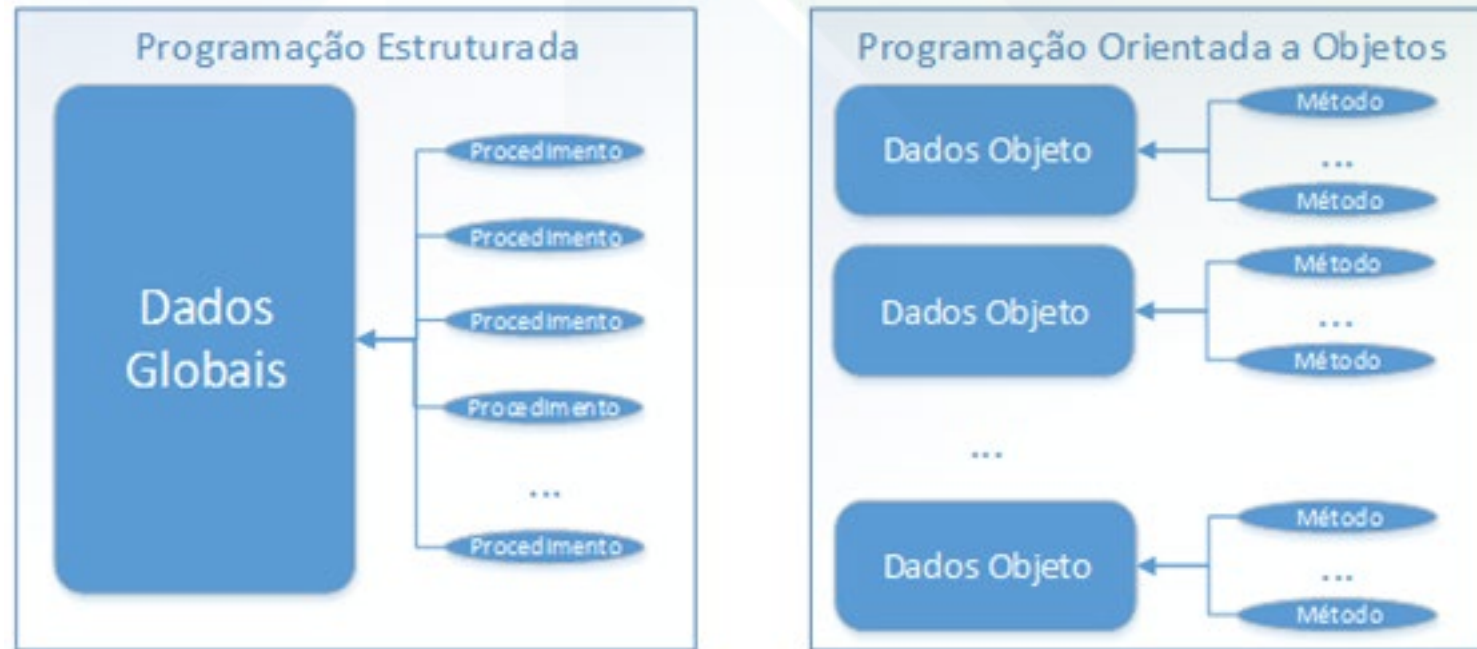
Com isso, é possível observar que a orientação a objetos não é recente; ao contrário, o conceito proposto de abstração de dados de uma estratégia ou regra de negócios para uma linguagem computacional permite, inclusive, aproveitar determinados códigos e rotinas de software.

Mas, afinal, o que é um objeto para a análise e desenvolvimento de sistemas? Lima (2011, p. 19) define objeto da seguinte forma: “[...] tudo o que é manipulável ou manufaturável; tudo o que é perceptível por qualquer dos sentidos; coisa, peça, artigo de compra e venda; matéria; assunto [...]”.

A partir da análise de um objeto, é possível visualizar com maior clareza o seu comportamento, bem como os seus atributos e características. Sommerville (2011) ressalta, ainda, que, quando se pensa em um projeto de software, essa ideia está intimamente interligada à sua implementação em uma linguagem de programação como, por exemplo, Java, Python, Ruby, C# etc.


Mas, antes de implementar, a análise de sistemas permite descrever os detalhes de um determinado objeto. Observe a Figura 9 que compara os paradigmas de programação estruturada e orientada a objetos:

Figura 9 – Paradigma de programação estruturada x orientada a objetos



Fonte: MACHADO, Henrique. Os quatro pilares da programação orientada a objetos. DevMedia. Disponível em: <<http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>>. Acesso em: 30 abr. 2016.

A Figura 9 ilustra como é concebido um projeto de software que se fundamenta no paradigma de programação estruturada, em que é possível considerar os elementos que provocarão ações do sistema como sendo de acesso global, com os respectivos procedimentos interligados



diretamente a ele. Na programação estruturada, prevalecem os procedimentos que estabelecem uma sequência, permitindo a seleção de ações e, ainda, de interação, através dos laços de repetição. No entanto, quando o programa começa a ficar mais complexo, é necessário dividi-lo em partes menores, que recebem o nome de módulos.

Já no paradigma da orientação a objetos, todos os métodos e procedimentos estão diretamente relacionados ao próprio objeto, de forma a manter o seu estado ou comportamento.

Algumas características da orientação a objetos podem ser destacadas, quando comparadas ao paradigma de programação estruturada. Observe o Quadro 7, a seguir:

Quadro 7 – Características de paradigmas de programação: estruturada x orientada a objetos

Programação orientada a objetos	Programação estruturada
Métodos	Procedimentos e funções
Instâncias de variáveis	Variáveis
Mensagens	Chamadas a procedimentos e funções
Classes	Tipos de dados definidos pelo usuário
Herança	Não disponível
Polimorfismo	Não disponível

Fonte: ABILIO, Igor. DevMedia. Programação Orientada a Objetos versus Programação Estruturada. Disponível em: <<http://www.devmedia.com.br/programacao-orientada-a-objetos-versus-programacao-estruturada/32813>>. Acesso em: 30 abr. 2016.

Enquanto a programação orientada a objetos considera os métodos de cada objeto, ou seja, as ações que ele realiza, a programação estruturada trabalha com procedimentos e funções que chamam as variáveis globais e que representam um determinado tipo de dado.





Leia o artigo “Programação orientada a objetos uma abordagem didática” e entenda quais são os conceitos relacionados a esse paradigma de programação. Disponível em: <[http://www.ccuec.unicamp.br/revista/infotec/artigos/leite\\_rahall.html](http://www.ccuec.unicamp.br/revista/infotec/artigos/leite_rahall.html)>. Acesso em: 5 maio 2016.

Além dessas, há ainda a possibilidade de se inserir os conceitos de encapsulamento e herança, que são comuns apenas ao paradigma de programação orientada a objetos. A seguir, neste material, serão descritos em detalhes os seus respectivos significados e exemplos.

Outro ponto distinto entre os paradigmas mencionados é que o uso de mensagens em OO é comum, enquanto que, em programação estruturada, há a chamada a procedimentos e funções. Além disso, em programação estruturada, são declaradas as variáveis. Em OO, utiliza-se o conceito de classes, ou seja, uma classe que pode conter mais de um objeto, inclusive com seus respectivos métodos.

Em função dessa facilidade é que esse paradigma ganhou um espaço no mundo dos negócios, pois permite acelerar, paralelamente a uma boa metodologia de gestão, o planejamento, a análise, o desenvolvimento e a implantação de um produto de software.





## Para saber mais

Fica a recomendação de leitura de fixação do capítulo 14 de:

DENNIS, Alan et al. **Análise e Projeto de Sistemas**. 5. ed. Rio de Janeiro: LTC, 2014.

### 1. Paradigma de Programação Orientada a Objetos

Para que possa identificar os elementos que o paradigma de programação orientada a objetos propõe, é necessário também que diante de um determinado contexto profissional, ou seja, de uma realidade do mercado de trabalho, o analista saiba identificar a partir de um exercício de abstração, de forma a

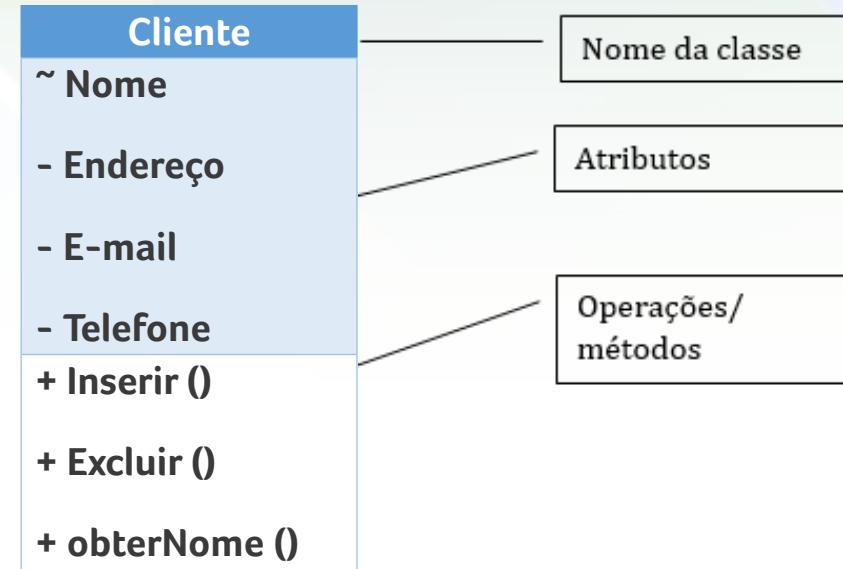
identificar os objetos desse sistema, quais serão as suas principais funções e ainda, quem são as pessoas que interagem com esta ação. Além disso, é preciso pensar também em como os dados provenientes dessa ação do sistema serão manipulados, armazenados e tratados.

Dessa forma, é preciso **modelar** esse sistema, ou seja, a partir da abstração de um determinado contexto, a sua representação por meio de um modelo é fundamental para facilitar a compreensão tanto por parte da equipe do projeto – analistas, desenvolvedores, gestores – quanto por parte do cliente, que deverá entender e aprovar as ações delimitadas em cada uma das fases do projeto.

Vamos agora compreender os principais conceitos inerentes ao paradigma de programação orientada a objetos. Nesse sentido, é preciso que se conheça o conceito de **classe**: “Uma classe é o modo geral que usamos para definir e criar instâncias específicas ou objetos. Cada objeto é associado a uma classe” (DENNIS; WIXON; ROTH, 2014, p. 491).

Observe a figura a seguir para a melhor compreensão dos conceitos, trabalharemos com uma classe chamada “Cliente”:

Figura 10 – Exemplo de classe em OO



Fonte: Elaborada pelo autor.

A classe cliente é comum a diversos tipos de projetos, por esse motivo, a sua escolha, então, a Figura 10 mostra exatamente o que representa uma classe em orientação a objetos, com os seus respectivos **atributos** que representam as propriedades ou



características da classe, e os métodos definidos ou lista de operações da classe.

Nem sempre é viável à aplicação que os atributos de uma classe sejam visíveis, então, nesse quesito, a sua visibilidade pode ser classificada como (LIMA, 2011): **pública (+)**, em que outras classes poderão ter acesso a esse atributo; **privada (-)**, em que o atributo pode ser utilizado e visto apenas pela própria classe; **protegida (#)**, em que o acesso se dá apenas pela própria classe e suas subclasses; **de pacote (~)**, que faz com que o atributo seja acessível pelas classes do pacote que a contém. Mas essa última questão de visibilidade é mais aplicada quando a linguagem de programação utilizada é Java.

Na sequência dessa apresentação das características de uma classe em orientação a objetos, vêm as *operações* da classe, ou seja, os **métodos** que definirão o seu comportamento (LIMA, 2011, p. 23):

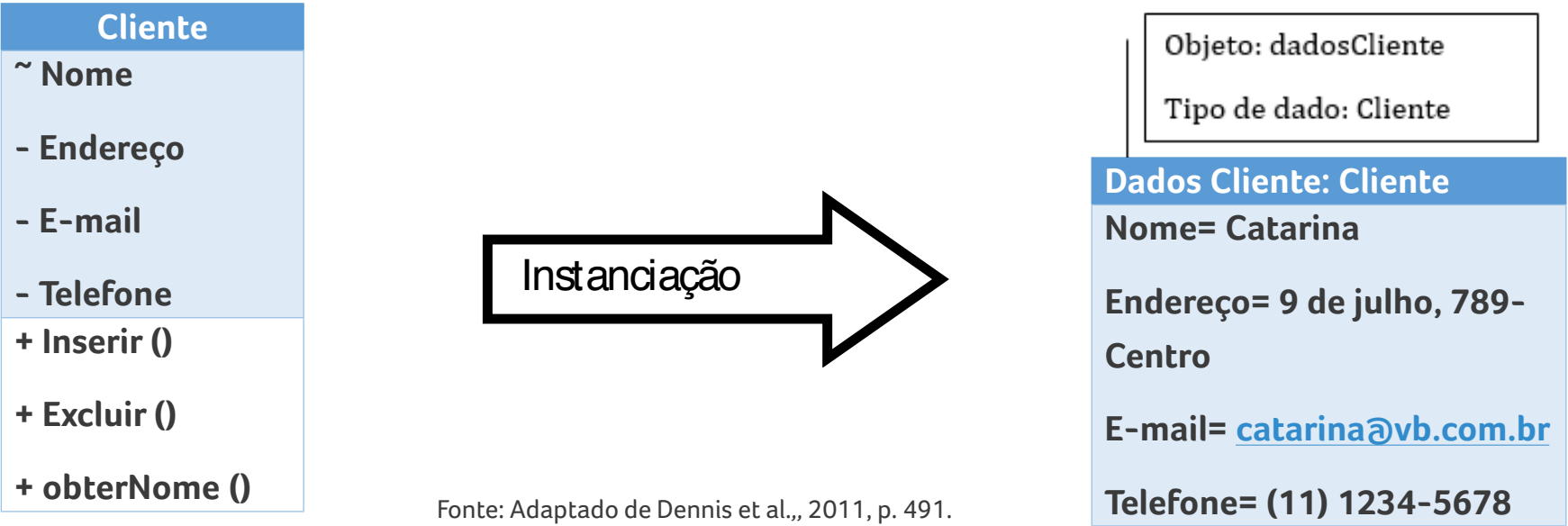


Para invocar um método de um objeto, envia-se uma mensagem para ele especificando o nome do objeto, o método a ser executado e a lista de argumentos requeridos. Após a execução, o objeto pode ou não retornar um valor como resposta à mensagem recebida.

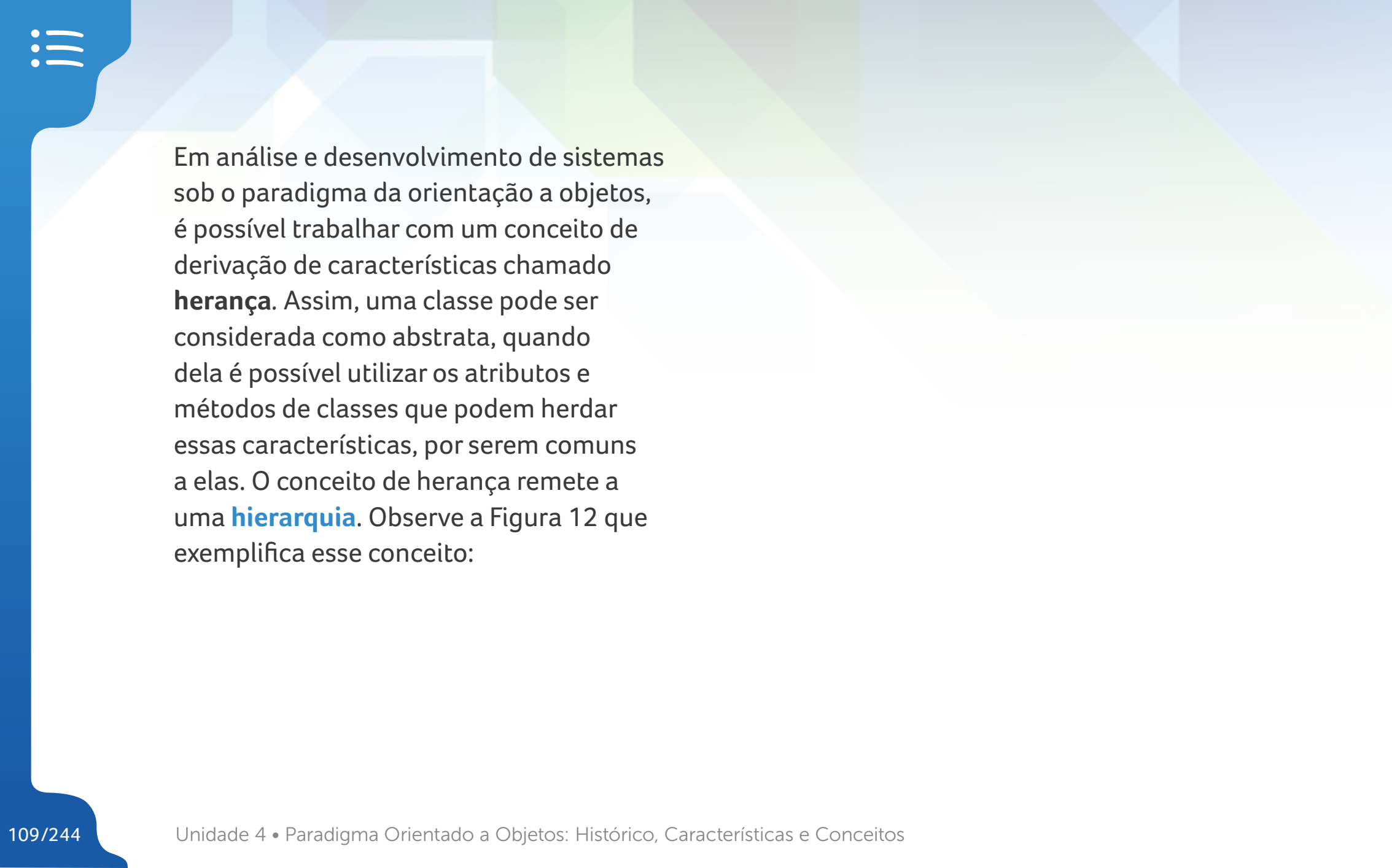
As operações também seguem os mesmos quesitos quanto à sua visibilidade. Outra característica de uma operação é que, além de um nome, possui a sua respectiva lista de argumentos e ainda o tipo de retorno que apresentará quando em execução. A essas definições de características de uma *classe* dá-se o nome de **assinatura de operação** (LIMA, 2011).

Uma classe também pode ser identificada pela classificação dos objetos a ela interligada, que têm o mesmo tipo de dado e comportamento. Nesse caso, cada objeto que pertence a uma classe é chamado de **instância de classe** (LIMA, 2011). Observe a Figura 11:

Figura 11 – Instância de classe

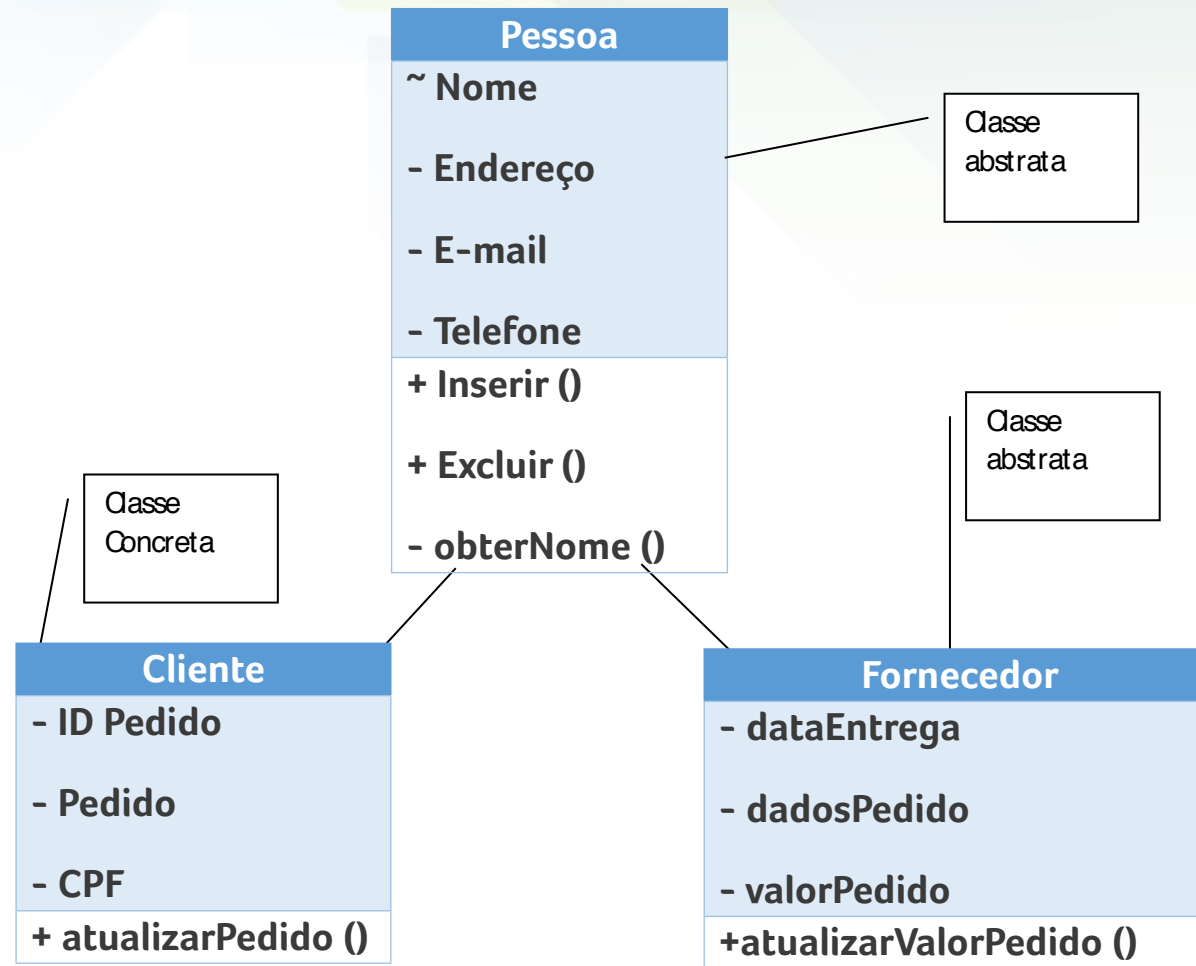


Fonte: Adaptado de Dennis et al., 2011, p. 491.




Em análise e desenvolvimento de sistemas sob o paradigma da orientação a objetos, é possível trabalhar com um conceito de derivação de características chamado **herança**. Assim, uma classe pode ser considerada como abstrata, quando dela é possível utilizar os atributos e métodos de classes que podem herdar essas características, por serem comuns a elas. O conceito de herança remete a uma **hierarquia**. Observe a Figura 12 que exemplifica esse conceito:

Figura 12 – Herança



Fonte: Adaptado de Dennis et al., 2011, p. 494.



Quando estabelece uma relação de herança, significa que os atributos da classe abstrata ou superclasse também são atributos das subclasses, no entanto essas também podem conter os seus próprios atributos e métodos, além dos herdados.


Outra característica é que uma classe concreta sempre terá objetos instanciados (DENNIS et al., 2014, p. 494):



A maioria das classes em toda hierarquia levará às instâncias, e qualquer classe que tenha instâncias é denominada uma classe concreta. [...] Algumas classes não produzem instâncias porque são usadas simplesmente como modelos para outras classes mais específicas (em especial localizadas em um nível alto de uma hierarquia). Elas são classes abstratas.



Assista à videoaula sobre herança, polimorfismo, encapsulamento e outros conceitos, disponível em:  
<<https://www.youtube.com/watch?v=LXCdObKZgo8>>. Acesso em: 6 maio 2016.



Em OO (Orientação a Objetos), também está fortemente presente o conceito de polimorfismo. Como o próprio nome diz, polimorfismo deriva de multifunções, multiforme, ou seja, que pode sofrer variações ou mesmo apresentar características distintas. Quando aplicado ao paradigma de programação orientada a objetos, pode ser definido, segundo Lima (2011, p. 24), da seguinte forma:



Polimorfismo é o princípio em que classes derivadas de uma mesma superclasse podem invocar operações que têm a mesma assinatura, mas comportamentos diferentes em cada subclasse, produzindo resultados diferentes, dependendo de como cada objeto implementa a operação.


Quando se aliam os conceitos de herança e polimorfismo, é preciso ter o cuidado de não **suprimir** da subclasse as características da superclasse.

### Para saber mais

No capítulo 1 do livro *UML 2.3 – Do requisito à solução*, você pode encontrar mais informações sobre herança, polimorfismo e encapsulamento.

LIMA, Adilson S. **UML 2.3** – Do requisito à solução. São Paulo: Érica, 2011.





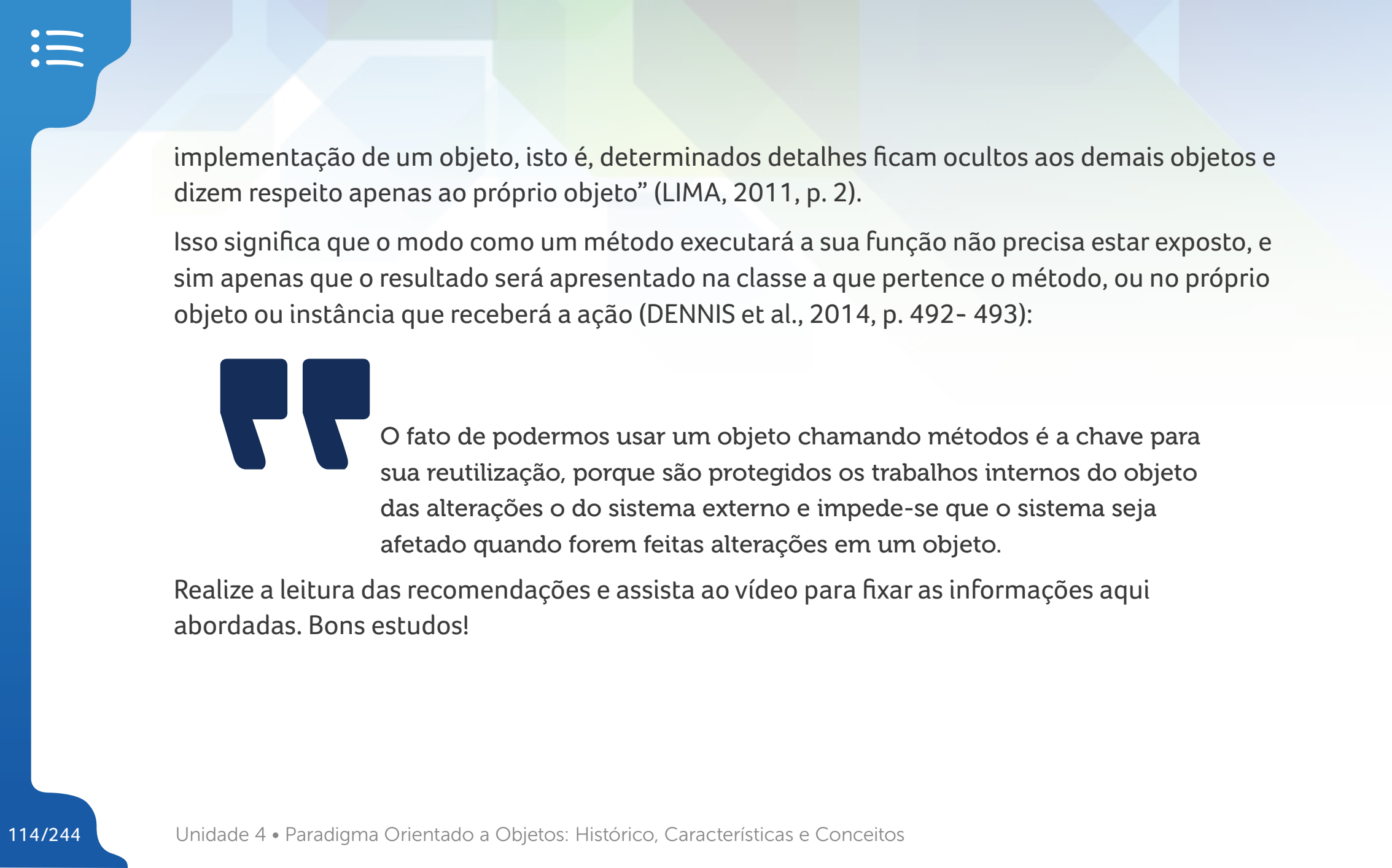
Por exemplo, o método “inserir”, que foi especificado na classe *Pessoa*, poderá ser utilizado, chamado a partir das subclasses *Cliente* e *Fornecedor*, porém, apesar do método conter o mesmo nome, o resultado será diferente e o armazenamento da informação também ocorrerá em locais diferentes. Isso significa que o comportamento mudou, apesar de ter o mesmo nome e ser utilizado em ambas subclasses. Se o método “*inserir*” for chamado a partir da classe *Fornecedor*, apesar de ser o mesmo método, assumirá um comportamento diferente, e será realizado o cadastro de um novo fornecedor, e assim por diante.

Link



Leia o artigo “As leis do mundo dos objetos” e aprenda mais formas de aplicação e utilização dos conceitos aqui estudados. Disponível em: <<http://www.devmedia.com.br/as-leis-do-mundo-dos-objetos-melhores-praticas-em-orientacao-a-objetos/26588>>. Acesso em: 5 maio 2016.

Por fim, para encerrar a apresentação dos conceitos pertinentes ao paradigma de programação orientada a objetos, é conceituado o **encapsulamento**: “[...] é uma técnica que consiste em separar aspectos externos dos internos da



implementação de um objeto, isto é, determinados detalhes ficam ocultos aos demais objetos e dizem respeito apenas ao próprio objeto” (LIMA, 2011, p. 2).

Isso significa que o modo como um método executará a sua função não precisa estar exposto, e sim apenas que o resultado será apresentado na classe a que pertence o método, ou no próprio objeto ou instância que receberá a ação (DENNIS et al., 2014, p. 492- 493):



O fato de podermos usar um objeto chamando métodos é a chave para sua reutilização, porque são protegidos os trabalhos internos do objeto das alterações o do sistema externo e impede-se que o sistema seja afetado quando forem feitas alterações em um objeto.

Realize a leitura das recomendações e assista ao vídeo para fixar as informações aqui abordadas. Bons estudos!



## Glossário

**Simula e Smaltalk:** linguagens de programação criadas na década de 1960 para o desenvolvimento de simulações de eventos em sistemas. *Simula* pode ser considerada a primeira linguagem de programação orientada a objetos e que implementava o conceito de herança.

**Suprimir:** eliminar, excluir uma informação ou objeto.

**Hierarquia:** estabelece nível, posição, autonomia de processos e compartilhamento de informações de acordo com a viabilidade da operação. Prioridade.



# Questão para reflexão

Diante das mudanças frequentes de requisitos por parte do *sponsor* do projeto, ou solicitante, de que forma a análise de sistema pode ser facilitada com o desenvolvimento do sistema sob o paradigma da programação orientada a objetos? Como esse processo está interligado a um método ágil de desenvolvimento?





## Considerações Finais

Nesta aula, você conheceu como se deu a evolução do processo de análise e desenvolvimento de sistemas atrelado à evolução do hardware e dos meios de armazenamento, de forma a promover o paradigma de programação orientada a objetos.

Você também foi apresentado aos principais conceitos relacionados à orientação a objetos, tais como: abstração, modelagem, classe, subclasse, classe abstrata, classe concreta, atributos e métodos.

Além dos conceitos mencionados, você conheceu de que forma os métodos podem ser utilizados em outras classes a partir da implementação de herança e polimorfismo.

Outro item importante relacionado à OO, o encapsulamento, foi mencionado para que você possa aprofundar os conhecimentos nesta técnica de ocultamento de informações e proteção a métodos.



## Referências

ABILIO, Igor. **DevMedia**. Programação Orientada a Objetos versus Programação Estruturada. Disponível em: <<http://www.devmedia.com.br/programacao-orientada-a-objetos-versus-programacao-estruturada/32813>>. Acesso em: 30 abr. 2016.

DENNIS, Alan et al. **Análise e Projeto de Sistemas**. 5. ed. Rio de Janeiro: LTC, 2014.

LIMA, Adilson S. **UML 2.3 do requisito a solução**. São Paulo: Érica, 2011.

MACHADO, Henrique. Os quatro pilares da programação orientada a objetos. DevMedia. Disponível em: <<http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>>. Acesso em: 30 abr. 2016.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.



# Assista a suas aulas



## **Aula 4 - Tema: Paradigma Orientado a Objetos: Histórico, Características e Conceitos - Bloco I**

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/88745c604fe820b3979424f33f410001>>.



## **Aula 4 - Tema: Paradigma Orientado a Objetos: Histórico, Características e Conceitos - Bloco II**

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/aa9a-59fcd92bdc4003be92365c0decfa>>.



# Questão 1

## 1. Analise as afirmações a seguir e assinale a alternativa correspondente:

I – O paradigma de programação orientada a objeto considera tanto dados como processos.

II – O paradigma de programação estruturada é centrado em dados ou em processos.

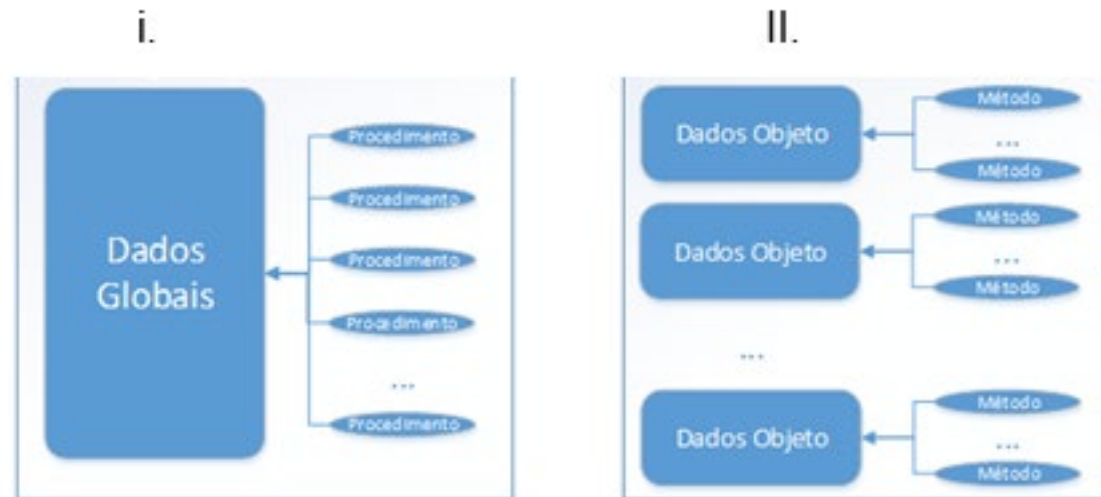
Quanto às afirmações, é correto afirmar que:

- a) as duas afirmações são verdadeiras;
- b) as duas afirmações são falsas;
- c) apenas a afirmação I é verdadeira;
- d) apenas a afirmação II é verdadeira;
- e) apenas a afirmação II é falsa.



## Questão 2

2. Observe as figuras que comparam os paradigmas de programação estruturada e orientada a objetos:



Fonte: MACHADO, Henrique. Os quatro pilares da programação orientada a objetos. DevMedia. Disponível em: <<http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>>. Acesso em: 30 abr. 2016.



## Questão 2

- a) As figuras I e II representam o paradigma de programação estruturada.
- b) As figuras I e II representam o paradigma de programação orientada a objetos.
- c) A figura I representa o paradigma de programação orientada a objetos.
- d) A figura II representa o paradigma de programação estruturada.
- e) As figuras I e II representam os paradigmas de programação estruturada e orientada a objetos respectivamente.



## Questão 3

3. A figura a seguir representa:

Cliente
~ Nome
- Endereço
- E-mail
- Telefone
+ Inserir ()
+ Excluir ()
+ obterNome ()

- a) Uma instância de cliente com atributo Nome com permissões de acesso de pacote e os demais atributos privados ao objeto.
- b) Uma classe com atributo Nome com permissões de acesso de pacote da classe e os demais atributos privados.



## Questão 3

- c) Uma herança de Classe com método de inserção com visibilidade privada.
- d) Um encapsulamento com método de exclusão com visibilidade pública.
- e) Um polimorfismo do atributo nome.



## Questão 4

**4. Complete as lacunas do trecho de texto abaixo, com os conceitos apresentados em uma das alternativas:**

“A maioria das classes em toda hierarquia levará às \_\_\_\_\_, e qualquer classe que tenha instâncias é denominada uma \_\_\_\_\_. [...] Algumas classes não produzem instâncias porque são usadas simplesmente como modelos para outras classes mais específicas (em especial localizadas em um nível alto de uma hierarquia). Elas são\_\_\_\_\_”.

- a) classe concreta / classes abstratas / instâncias
- b) classe / classes abstratas / instâncias
- c) instâncias / classe concreta / classes abstratas.
- d) classes abstratas / instâncias / instâncias
- e) classes abstratas / classe concreta / classe



## Questão 5

### 5. Analise as afirmações:

I – “Princípio em que classes derivadas de uma mesma superclasse podem invocar operações que têm a mesma assinatura, mas comportamentos diferentes em cada subclasse, produzindo resultados diferentes, dependendo de como cada objeto implementa a operação” (LIMA, 2011, p. 24).

II – “[...] é uma técnica que consiste em separar aspectos externos dos internos da implementação de um objeto, isto é, determinados detalhes ficam ocultos aos demais objetos e dizem respeito apenas ao próprio objeto” (LIMA, 2011, p. 2).

Os conceitos apresentados são, respectivamente, referentes a(à):

- a) classes abstratas e encapsulamento;
- b) classes concretas e encapsulamento;
- c) herança e encapsulamento;
- d) polimorfismo e encapsulamento;
- e) encapsulamento e polimorfismo.



# Gabarito

## 1. Resposta: A.

Comentário: em OO, os dados e os processos precisam ser analisados, direcionados e implementados de acordo com as diretrizes especificadas de forma a não alterar as características dos objetos e métodos.

## 2. Resposta: E.

Comentário: a figura ilustra como é concebido um projeto de software que se fundamenta no paradigma de programação estruturada, em que é possível considerar os elementos que provocarão ações do sistema como sendo de acesso global, com os respectivos

procedimentos interligados diretamente a ele. Já no paradigma da orientação a objetos, todos os métodos e procedimentos estão diretamente relacionados ao próprio objeto, de forma a manter o seu estado ou comportamento.

## 3. Resposta: B.

Comentário: apenas a alternativa B está correta. As demais não são aplicáveis segundo os paradigmas de programação e lógica computacional.

## 4. Resposta: C.

“A maioria das classes em toda hierarquia levará às instâncias, e qualquer classe que tenha instâncias é denominada uma



# Gabarito

classe abstrata. [...] Algumas classes não produzem instâncias porque são usadas simplesmente como modelos para outras classes mais específicas (em especial localizadas em um nível alto de uma hierarquia). Elas são classes concretas” (DENNIS et al., 2014, p. 494).

precisa estar exposto, e sim apenas que o resultado será apresentado na classe a que pertence o método, ou no próprio objeto ou instância que receberá a ação.

## 5. Resposta: D.

Comentário: em OO (Orientação a Objetos), está fortemente presente o conceito de **polimorfismo** que deriva de multifunções, é multiforme, ou seja, que pode sofrer variações ou mesmo apresentar características distintas. Já o **encapsulamento** representa o modo como um método executará a sua função. Não





# Unidade 5

A Linguagem de Modelagem Unificada – Unified Modeling Language (UML): Histórico e Visão Geral das Técnicas de Modelagem. IFML

---

## Objetivos

Com esta disciplina, temos os objetivos de:

1. Apresentar a análise de sistemas sob o ponto de vista da orientação a objetos.
2. Associar a utilização da linguagem de modelagem unificada (UML) com a orientação a objetos, de forma a permitir maior detalhamento e visibilidade das ações do sistema.
3. Abordar os principais diagramas considerados em UML para auxiliar no processo de análise e desenvolvimento de sistemas.



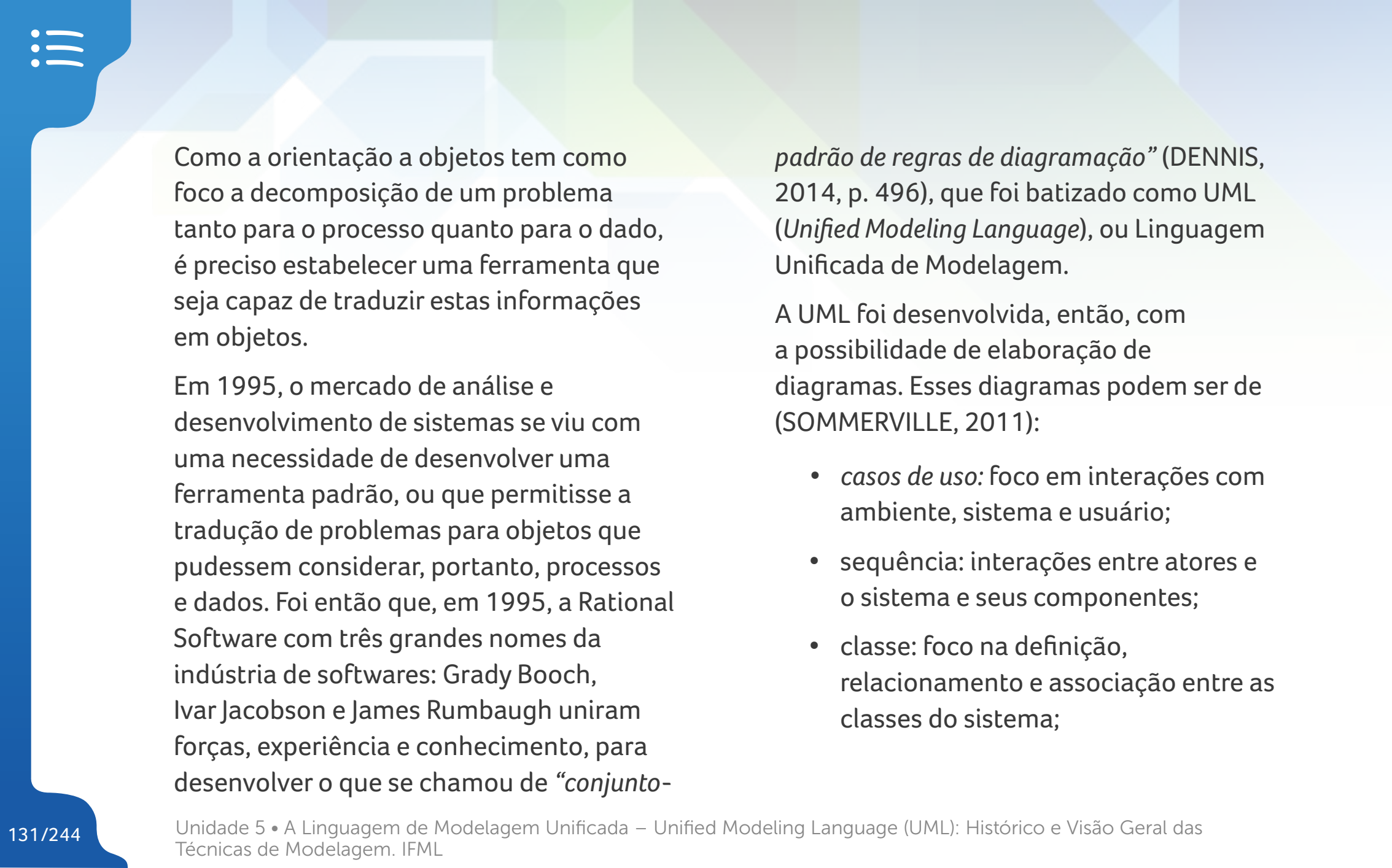
## Introdução

Esta aula apresenta os conceitos relacionados à modelagem de sistema pela ótica da linguagem de modelagem unificada, também conhecida como UML, para facilitar e apoiar os processos de análise e desenvolvimento de sistemas que seguem as **premissas** do paradigma de programação orientada a objetos (SOMMERVILLE, 2011, p. 125):



Sistemas orientados a objetos são mais fáceis de mudar do que os sistemas desenvolvidos com abordagens funcionais. Os objetos incluem os dados e as operações para manipulá-los. Portanto, eles podem ser entendidos e modificados como entidades autônomas.

Nesse sentido, também vale ressaltar que, para gerir as equipes de análise e desenvolvimento de sistemas, é bastante recomendável a adoção de um método ágil, pois, aliados às técnicas da orientação a objetos e à modelagem de sistema a partir da implementação da UML, os resultados podem ser melhores do que aqueles observados com os modelos tracionais de análise e desenvolvimento de sistemas: "A diferença principal entre uma abordagem tradicional, como o design estruturado, e uma abordagem orientada a objetos é como um problema é decomposto" (DENNIS, 2014, p. 495).



Como a orientação a objetos tem como foco a decomposição de um problema tanto para o processo quanto para o dado, é preciso estabelecer uma ferramenta que seja capaz de traduzir estas informações em objetos.

Em 1995, o mercado de análise e desenvolvimento de sistemas se viu com uma necessidade de desenvolver uma ferramenta padrão, ou que permitisse a tradução de problemas para objetos que pudessem considerar, portanto, processos e dados. Foi então que, em 1995, a Rational Software com três grandes nomes da indústria de softwares: Grady Booch, Ivar Jacobson e James Rumbaugh uniram forças, experiência e conhecimento, para desenvolver o que se chamou de “conjunto-

padrão de regras de diagramação” (DENNIS, 2014, p. 496), que foi batizado como UML (*Unified Modeling Language*), ou Linguagem Unificada de Modelagem.

A UML foi desenvolvida, então, com a possibilidade de elaboração de diagramas. Esses diagramas podem ser de (SOMMERVILLE, 2011):

- *casos de uso*: foco em interações com ambiente, sistema e usuário;
- *sequência*: interações entre atores e o sistema e seus componentes;
- *classe*: foco na definição, relacionamento e associação entre as classes do sistema;


- estado: visa demonstrar como ocorre a mudança de estados em função de eventos internos ou externos ao sistema.

Segundo os seus desenvolvedores, quando se trabalha com um projeto orientado a objetos, o produto de software deve ser dirigido a casos de uso, centrado na arquitetura, ser interativo e, ao mesmo tempo, incremental.

Uma modelagem de sistema baseada em **casos de uso** permite que se identifique qual será o comportamento do sistema:



Um caso de uso descreve como um usuário interage com o sistema para realizar alguma atividade, como emitir um pedido, fazer uma reserva ou procurar por uma informação. Os casos de uso têm por função identificar e comunicar os requisitos do sistema aos programadores que devem fazê-lo (DENNIS, 2014, p. 496).





Com os casos de uso, utilizados em projetos de software orientados a objetos, a tarefa de decompor os processos se tornou mais simples. Cada atividade do sistema será representada através de um diagrama de caso de uso, dessa forma facilitando a visão do comportamento desta ação no sistema, e, conseqüentemente, a sua programação: “As abordagens orientadas a objeto concentram-se em focar apenas uma atividade de caso de uso por vez e distribuir esse caso de uso em um conjunto de objetos que se comunicam e colaboram entre si” (DENNIS, 2014, p. 496).



Aprenda sobre IFML (Interaction Flow Modeling Language) e como essa linguagem de modelagem da interação do usuário com o sistema pode contribuir para com os projetos. Artigo disponível em: <<http://www.devmedia.com.br/trabalhando-com-interaction-flow-modeling-language-ifml/33793>>. Acesso em: 15 maio 2016.

No entanto, não são apenas os processos que devem ser decompostos; a UML pode auxiliar no projeto de modelagem do sistema, ou seja, permitir que se identifique a sua arquitetura. Com isso, é possível dizer que a análise e o projeto de interface também fazem parte da arquitetura,



então, Dennis et al. (2014, p. 496) dizem que esse tipo de modelagem de sistema é “centrado na arquitetura”. Isso quer dizer que serão identificadas as especificações do projeto, a sua análise e desenvolvimento e, também, a documentação do sistema: “Uma vez definidas as interações entre o sistema de software e o ambiente do sistema, você pode usar essa informação como base para projetar a arquitetura do sistema” (SOMMERVILLE, 2011, p. 127).

Quando o enfoque está na arquitetura, essa pode ser classificada como:

- **funcional:** esse enfoque auxilia na identificação do comportamento do sistema, de acordo com a visão do usuário.
- **estático:** esse define quais são as características da estrutura do sistema no que tange a identificação das classes, bem como de seus atributos, métodos, as mensagens que deverão ser trocadas entre as classes e os seus respectivos relacionamentos.
- **dinâmico:** esse modo descreve tanto a mudança de estado dos objetos quanto as mensagens que são trocadas entre eles.



Aprenda mais sobre IFML e descubra as possibilidades trazidas no design de interface e as interações com o usuário. Minimize erros. Aumente as boas experiências no uso de software. Artigo disponível em: <<http://www.omg.org/spec/IFML/>>. Acesso em: 15 maio 2016.

Além desses modos de se fazer a modelagem de um sistema, existe ainda o **iterativo ou incremental**, que considera basicamente os testes que devem ser realizados a cada fase do desenvolvimento do sistema dessa forma, portanto depende do ciclo de vida do projeto.

## 1. UML

A UML tem sido aplicada em projetos a fim de estabelecer uma forma de representação que permita modelar o sistema através de um vocabulário que seja comum à programação orientada a objetos. Em resumo, a UML, já na versão 2.0, segundo a aprovação da OMG (Object Management Group), apresenta 14 tipos de análises por diagramas para fazer a modelagem de sistemas.

## Para saber mais

Leia o capítulo 2 do livro de MELO, Ana Cristina. **Desenvolvendo aplicações com UML 2.2: do conceitual à implementação**. 3. ed. Rio de Janeiro: Brasport, 2010.

São classificados em **diagramas estruturais**: “[...] usados na representação dos dados e dos relacionamentos estáticos que existem em um sistema de informação” (DENNIS et al., 2014, p. 498), e os **diagramas de comportamento**: “[...] fornecem ao analista uma maneira de representar os relacionamentos dinâmicos entre as instâncias ou objetos [...]”. (DENNIS et al., 2014, p. 498).

Veja nos quadros abaixo os tipos de diagramas:

Quadro 8 – Diagramas estruturais

Diagramas Estruturais	Descrição
Classe	Apresenta as classes e as suas relações diante do sistema. É realizado na fase de análise e design de interface.
Objeto	Apresenta os relacionamentos estabelecidos entre os objetos do sistema. É realizado na fase de análise e design de interface.



Diagramas Estruturais	Descrição
Pacote	Representa várias funções do sistema através de UML para representar a interligação de algumas ações, ou seja, classes, métodos e objetos que se correlacionam. Amplamente utilizado nas fases de análise, design e implementação.
Utilização	Foco em apresentar a estrutura física do sistema. É realizado nas fases de design físico e implementação.
Componente	Mostra os relacionamentos entre os componentes do sistema. Parte física. É realizado nas fases de design físico e implementação.
Estrutura composta	Amplamente utilizado nas fases de análise e design, pois apresenta os relacionamentos entre os elementos de uma classe e seus relacionamentos.

Fonte: Adaptado de Dennis, 2014, p. 499.

*Para saber mais*

Leia o capítulo 1 do livro de GUEDES, Gilleanes T.A. **UML 2: guia prático**. 2. ed. São Paulo: Novatec, 2014.

A seguir, o Quadro 9 apresenta os diagramas de comportamento:

Quadro 9 – Diagramas de comportamento

Diagramas de Comportamento	Descrição
Atividade	Foca o fluxo de trabalho em um caso de uso. Representa as regras de negócios. Utilizado em análise e design.
Sequência	Representa o comportamento dos objetos. Pode ser elaborado com casos de uso. Ordena as atividades. Utilizado em análise e design.
Comunicação	Mostra o comportamento de um objeto através do caso de uso. Foca a colaboração entre objetos em uma determinada ação do sistema, ou atividade. Utilizado em análise e design.
Visão geral da interação	Fluxo de controle do processo. Utilizado em análise e design.
Estado comportamental/ Protocolo	Mostra o comportamento de uma classe e o seu respectivo vínculo ou dependências com outras interfaces de classes. É utilizado em análise e design.
Caso de Uso	Mostra a interação entre o sistema e ambiente, entre o usuário e o sistema e entre módulos do sistema. Segue os requisitos do negócio e é utilizado na fase de análise.

Fonte: Adaptado de Dennis, 2014, p. 499.



Estude e aprenda desde já como você pode trabalhar com os diagramas de caso de uso. Artigo disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409432.aspx>>. Acesso em: 15 maio 2016.

Apesar de fornecer subsídios para a modelagem de sistemas com termos comuns à orientação a objetos, UML não é uma metodologia, mas sim pode ser trabalhada juntamente com uma metodologia de análise e desenvolvimento de sistema. Uma das mais conhecidas é RUP (Rational Unified Process) ou Racional Unificado. A seguir, veremos mais informações sobre essa metodologia.

## 2 RUP (Rational Unified Process)

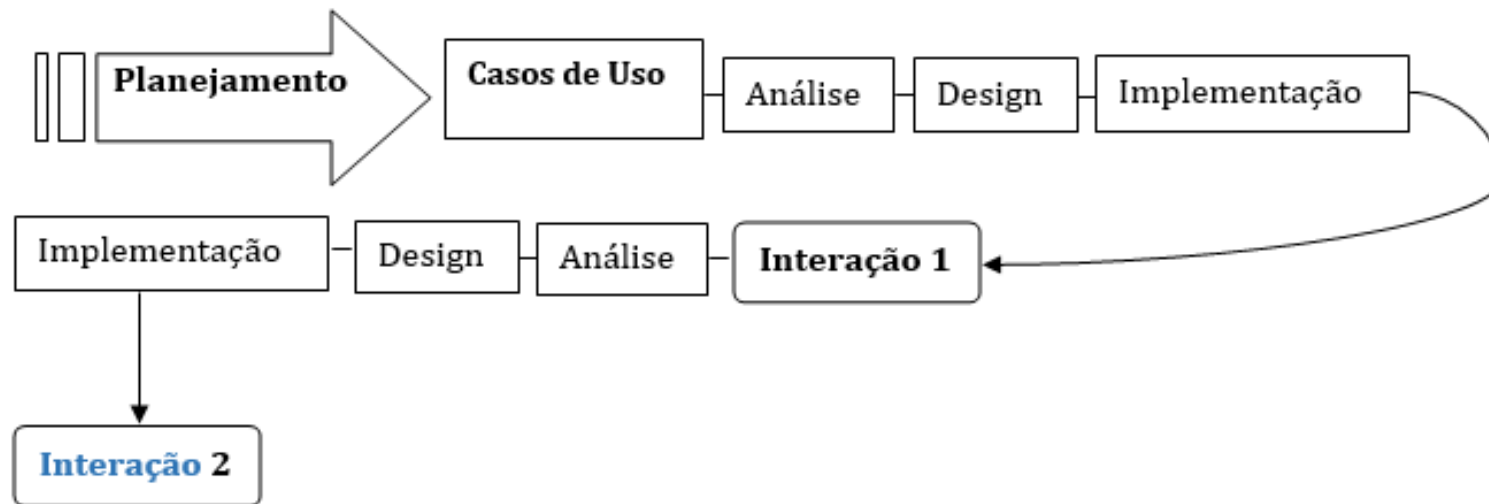
Essa é uma metodologia que foi desenvolvida por uma empresa denominada Rational Software Corporation para aplicar os diagramas UML. Segue as premissas do desenvolvimento rápido, assim como XP e *Scrum*, previamente estudadas, mas, como considera basicamente a aplicação UML, se adéqua mais a esta etapa de estudos.

## Para saber mais


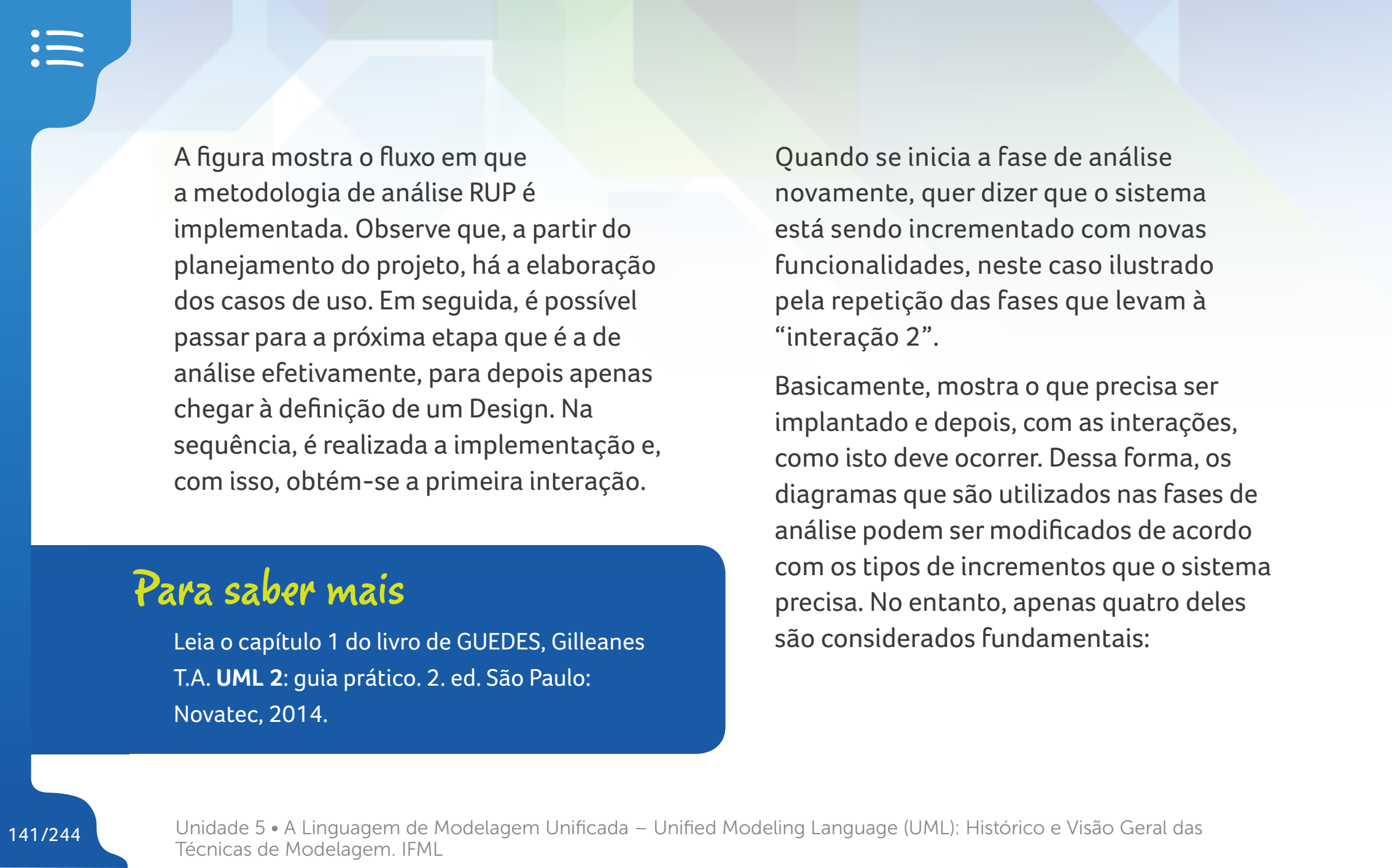
Sobre RUP, aprenda mais sobre como aplicar, sobre o diferencial que pode trazer para os projetos. Artigo disponível em: <<http://www.tiespecialistas.com.br/2011/02/rup-primeiros-passos/>>. Acesso em: 15 maio 2016.

Observe o fluxograma abaixo da lógica empregada na metodologia RUP:

Figura 13 – RUP aplicado à UML para análise e modelagem de sistemas



Fonte: Adaptado de Dennis et al., 2014.



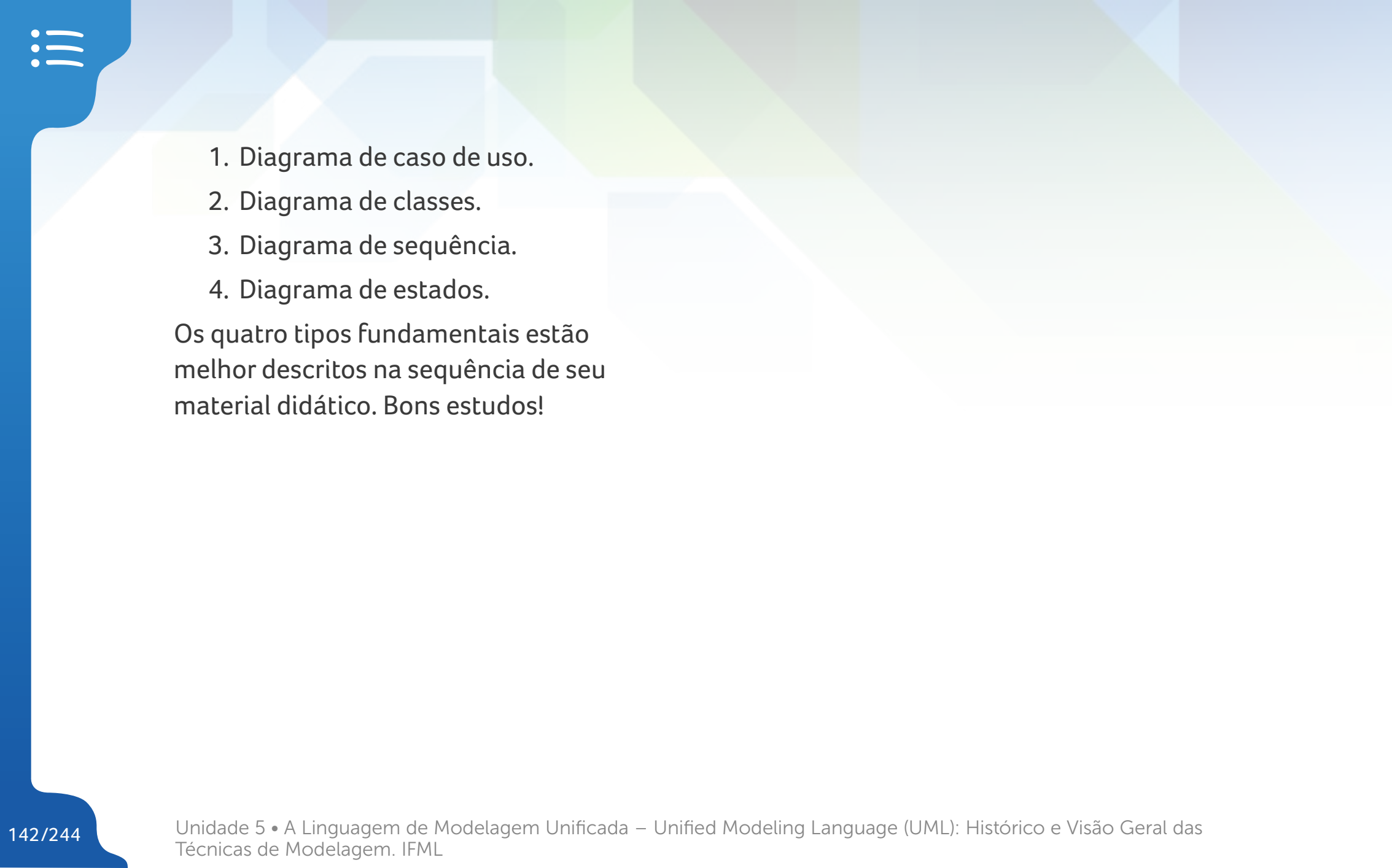

A figura mostra o fluxo em que a metodologia de análise RUP é implementada. Observe que, a partir do planejamento do projeto, há a elaboração dos casos de uso. Em seguida, é possível passar para a próxima etapa que é a de análise efetivamente, para depois apenas chegar à definição de um Design. Na sequência, é realizada a implementação e, com isso, obtém-se a primeira interação.

Quando se inicia a fase de análise novamente, quer dizer que o sistema está sendo incrementado com novas funcionalidades, neste caso ilustrado pela repetição das fases que levam à “interação 2”.

Basicamente, mostra o que precisa ser implantado e depois, com as interações, como isto deve ocorrer. Dessa forma, os diagramas que são utilizados nas fases de análise podem ser modificados de acordo com os tipos de incrementos que o sistema precisa. No entanto, apenas quatro deles são considerados fundamentais:

## Para saber mais

Leia o capítulo 1 do livro de GUEDES, Gilleanes T.A. **UML 2: guia prático**. 2. ed. São Paulo: Novatec, 2014.

- 
- 
1. Diagrama de caso de uso.
  2. Diagrama de classes.
  3. Diagrama de sequência.
  4. Diagrama de estados.

Os quatro tipos fundamentais estão melhor descritos na sequência de seu material didático. Bons estudos!



## Glossário

**Premissas:** essência, fundamento, ponto de partida.

**Interface:** que se conecta ou comunica com partes. Que faz o intermédio normalmente entre usuários e sistemas.

**Interação:** ação entre duas classes ou atividades que dependem ou se comunicam em um sistema.



# Questão para reflexão

Como é possível integrar métodos de desenvolvimento ágil com a gestão de projetos e a modelagem de sistema com UML? De que forma esses conceitos integrados e trabalhados em conjunto favorecem um projeto de software?







## Considerações Finais

Neste tema, você pode recuperar alguns conceitos relacionados à UML para modelagem de sistemas, além de poder associar às técnicas de desenvolvimento ágil, porém aplicadas à modelagem de sistema com a metodologia RUP.

Você foi apresentado aos tipos de diagramas estruturais e comportamentais que fundamentam a modelagem de sistemas através da UML, podendo identificar nos quadros a que fase podem ser aplicados cada um dos diagramas UML e o que representam.



## Referências

DENNIS, Alan et al. **Análise e Projeto de Sistemas**. 5. ed. Rio de Janeiro: LTC, 2014.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.



# Assista a suas aulas



## Aula 5 - Tema: A Linguagem de Modelagem Unificada - Bloco I

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/ab4d76c7aa70f8025408083be7ce5e41>>.



## Aula 5 - Tema: A Linguagem de Modelagem Unificada - Bloco II

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/582816af854db993433e033d4c99d3b8>>.



# Questão 1

## 1. Sobre as afirmações que seguem:

I – Casos de uso: foco em interações com ambiente, sistema e usuário.

II – Sequência: interações entre atores e o sistema e seus componentes.

III – Classe: foco na definição, relacionamento e associação entre as classes do sistema.

IV – Estado: visa demonstrar como ocorre a mudança de estados em função de eventos internos ou externos ao sistema.

- a) As afirmações são verdadeiras.
- b) As afirmações são falsas.
- c) Apenas a afirmação III é falsa.
- d) As afirmações I e II apenas são verdadeiras.
- e) ) Apenas a afirmação IV é falsa.



## Questão 2

**2. Funcional:** esse enfoque auxilia na identificação do comportamento do sistema, de acordo com a visão do usuário. **Estático:** esse define quais são as características da estrutura do sistema no que tange à identificação das classes, bem como de seus atributos, métodos, as mensagens que deverão ser trocadas entre as classes e os seus respectivos relacionamentos. **Dinâmico:** esse modo descreve tanto a mudança de estado dos objetos quanto as mensagens que são trocadas entre eles.

Referem-se a(à):

- a) casos de uso;
- b) orientação a objetos;
- c) modelagem da arquitetura do sistema;
- d) modelagem das classes e objetos;
- e) metodologia RUP.



## Questão 3

### 3. Os diagramas descritos abaixo são, respectivamente:

Classe	Apresenta as classes e as suas relações diante do sistema. É realizado na fase de análise e design de interface.
Objeto	Apresenta os relacionamentos estabelecidos entre os objetos do sistema. É realizado na fase de análise e design de interface.

- a) Comportamental, estrutural.
- b) Estrutural, comportamental.
- c) Comportamental, comportamental.
- d) Estrutural, estrutural.
- e) Dinâmico, estrutural.



# Questão 4

## 4. Associe os conceitos no quadro:

Diagramas de Comportamento	Descrição
I – Atividade	( ) Representa o comportamento dos objetos. Pode ser elaborado com casos de uso. Ordena as atividades. Utilizado em análise e design.
II – Sequência	( ) Foca o fluxo de trabalho em um caso de uso. Representa as regras de negócios. Utilizado em análise e design.
III – Comunicação	( ) Mostra o comportamento de um objeto através do Caso de Uso. Foca a colaboração entre objetos em uma determinada ação do sistema, ou atividade. Utilizado em análise e design.
IV – Visão geral da interação	( ) Fluxo de controle do processo. Utilizado em análise e design.
V – Estado comportamental/ protocolo	( ) Mostra a interação entre o sistema e ambiente, entre o usuário e o sistema e entre módulos do sistema. Segue os requisitos do negócio e é utilizado na fase de análise.
VI – Caso de uso	( ) Mostra o comportamento de uma classe e o seu respectivo vínculo ou dependências com outras interfaces de classes. Utilizado em análise e design.



## Questão 4

Representa a sequência correta da associação:

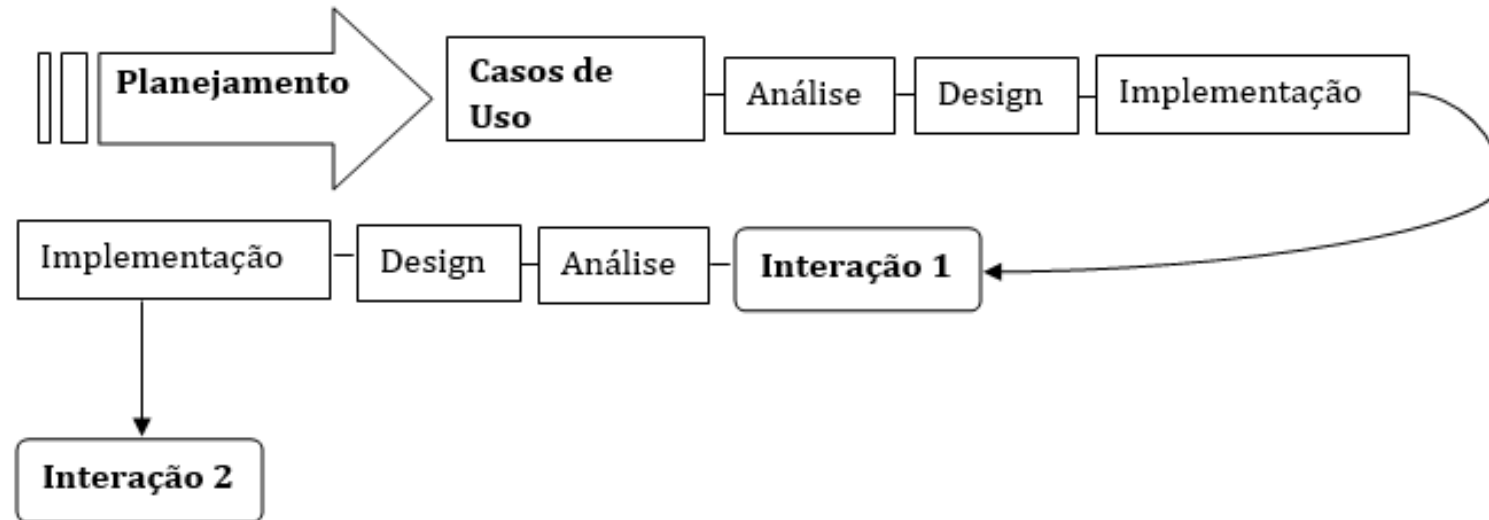
- a) I, II, III, IV, V, VI.
- b) II, I, III, IV, VI, V.
- c) V, VI, IV, III, II, I.
- d) III, V, II, I, IV, VI.
- e) IV, VI, V, I, II, III.





## Questão 5

5. Analise a figura abaixo e assinale a alternativa correspondente à metodologia de análise que está sendo empregada:



- a) Extreme Programming
- b) Scrum
- c) UML
- d) RUP
- e) SOA



# Gabarito

## 1. Resposta: A.

Comentário: estão descritos os quatro diagramas fundamentais da orientação a objetos, e corretamente. Portanto, todas as afirmações são verdadeiras.

## 2. Resposta: C.

Comentário: centrado na arquitetura: isso quer dizer que serão identificadas as especificações do projeto, a sua análise e desenvolvimento, e, também, a documentação do sistema: “Uma vez definidas as interações entre o sistema de software e o ambiente do sistema, você pode usar essa informação como base para projetar a arquitetura do sistema” (SOMMERVILLE, 2011, p. 127).

## 3. Resposta: D.

Comentário: são classificados em diagramas estruturais: “[...] usados na representação dos dados e dos relacionamentos estáticos que existem em um sistema de informação” (DENNIS et al., 2014, p. 498), e os diagramas de comportamento: “[...] fornecem ao analista uma maneira de representar os relacionamentos dinâmicos entre as instâncias ou objetos [...]” (DENNIS et al., 2014, p. 498).



## 4. Resposta: B.

Comentário:

Diagramas de Comportamento	Descrição
I – Atividade	( II ) Representa o comportamento dos objetos. Pode ser elaborado com casos de uso. Ordena as atividades. Utilizado em análise e design.
II – Sequência	( I ) Foca o fluxo de trabalho em um caso de uso. Representa as regras de negócios. Utilizado em análise e design.
III – Comunicação	( III ) Mostra o comportamento de um objeto através do Caso de Uso. Foca a colaboração entre objetos em uma determinada ação do sistema, ou atividade. Utilizado em análise e design.
IV – Visão geral da interação	( IV ) Fluxo de controle do processo. Utilizado em análise e design.
V – Estado comportamental/ protocolo	( V ) Mostra a interação entre o sistema e ambiente, entre o usuário e o sistema e entre módulos do sistema. Segue os requisitos do negócio e é utilizado na fase de análise.
VI – Caso de uso	( ) Mostra o comportamento de uma classe e o seu respectivo vínculo ou dependências com outras interfaces de classes. Utilizado em análise e design.



## 5. Resposta: C.

Comentário: A figura mostra o fluxo em que a metodologia de análise RUP é implementada. Observe que, a partir do planejamento do projeto, há a elaboração dos casos de uso. Em seguida, é possível passar para a próxima etapa, que é a de análise efetivamente, para depois apenas chegar à definição de um Design. Na sequência, é realizada a implementação e, com isso, obtém-se a primeira interação.



# Unidade 6

Modelo de Use Cases: Diagrama de Use Cases e Documentação de Use Cases. Especificação das Técnicas de Modelagem com Ferramenta Case

---

## Objetivos

Com esta disciplina, temos os objetivos de:

1. Conhecer e saber aplicar o diagrama de caso de uso de acordo com as regras de negócios do projeto a que se aplica.
2. Saber implementar a modelagem sob a visão do paradigma orientado a objetos e a sua abstração através do uso de UML com os diagramas de caso de uso.
3. Analisar e avaliar se o diagrama desenvolvido atende às premissas do projeto e aos requisitos de sistema, de forma a representar as interações que este deverá conter, integralmente.

## Introdução

A introdução dos diagramas apoiando a análise e desenvolvimento de sistemas muito tem contribuído para com a questão do nível de qualidade do produto de software que se pretende entregar ao cliente. Os diagramas não apenas permitem demonstrar as interações, sequência de ações, a estrutura do sistema, o estado, como também auxiliam na minimização de **incidência** de erros nos processos de análise e desenvolvimento de sistemas.

Isso se dá em função do uso da padronização trazida com a UML, que permitiu a inserção de tais representações gráficas, que por meio da abstração do mundo das regras de negócios, até a sua transposição a uma linguagem

computacional, podem representar e permitir a visualização de todo o sistema.

Com isso, um dos primeiros diagramas que você precisa conhecer é o de casos de uso. Segundo Dennis (2014), esse diagrama permite representar a finalidade de um determinado módulo ou funcionalidade de sistema incrementada, de forma a conseguir expressar, pela diagramação UML, o que o sistema deve fazer. Para isso, é preciso realizar um bom levantamento de requisitos; dessa forma, haverá subsídios para o desenvolvimento de uma boa documentação:




Um caso de uso pode representar diversos “caminhos” que é possível ao usuário percorrer ou interagir com o sistema; cada caminho é denominado “cenário”. Os casos de uso e os diagramas de caso de uso fornecem o suporte à visão funcional já descrita (DENNIS, 2014, p. 501).

Com os casos de uso, é possível descrever as interações que ocorrerão no sistema, ou seja, ações entre usuários e sistema, bem como a interação entre sistemas. O modo de representação de um caso de uso é bem simples. Considera a forma geométrica elipse para representar a ação, e atores, para indicar quem envia e quem recebe a solicitação. Além desses elementos, também são utilizadas as setas, que indicam o sentido do fluxo do processo.

Figura 14 – Exemplo de caso de uso em um sistema de cadastro



Fonte: Adaptado de Sommerville, 2011, p. 86.



Observe que fica mais fácil compreender a ação do sistema, e como está interagindo com o ambiente externo. Várias formas de representação podem ser utilizadas com os casos de uso:



Diagramas de caso de uso dão uma visão simples de uma interação. Logo, é necessário fornecer mais detalhes para entender o que está envolvido. Esses detalhes podem ser uma simples descrição textual, uma descrição estruturada em uma tabela ou um diagrama de sequência [...]. Você deve escolher o formato mais adequado, dependendo do caso de uso, e o nível de detalhamento que você acredita ser necessário no modelo (SOMMERVILLE, 2011, p. 87).

Com isso, você consegue associar ao processo de análise uma metodologia de desenvolvimento ágil. Dessa forma, as descrições podem ser melhor aproveitadas para aumentar a produtividade da equipe e maximizar a qualidade do produto de software. Assim, as teorias convergem e a prática ganha em resultados.





Link

Leia o material didático de José Carlos Macoratti, e aprenda mais sobre as especificações de casos de uso. MACORATTI, José Carlos. Modelando sistemas em UML: casos de uso. Disponível em: <[http://www.macoratti.net/net\\_uml2.htm](http://www.macoratti.net/net_uml2.htm)>. Acesso em: 16 maio 2016.

Uma das ferramentas que tem sido amplamente utilizadas, tanto no ambiente acadêmico quanto em projetos profissionais, é o Astah. Traz um ambiente prático e relativamente simples para desenvolver qualquer um dos diagramas UML. Também tem diversos materiais disponíveis para auxiliar na compreensão e uso da ferramenta.

Unidade 6 • Modelo de Use Cases: Diagrama de Use Cases e Documentação de Use Cases. Especificação das Técnicas de Modelagem com Ferramenta Case



Link

Aprenda mais sobre o Astah. Disponível em: <<http://astah.net/editions/community>>. Acesso em: 16 maio 2016.

A partir de agora, você conhecerá outros aspectos dos diagramas de caso uso, tais como: a identificação dos casos de uso, os atores, e também como inserir casos de uso em diagramas. Bons estudos!

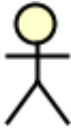
## 1. Casos de Uso


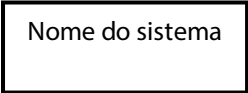

Os casos de uso compõem os **diagramas** de caso de uso. Isso porque se recomenda o uso de aproximadamente oito casos de uso por projeto. Por isso, os diagramas de caso

de uso podem servir como uma espécie de resumo de casos de uso que temos à disposição no projeto. Além disso, mostram-se como ótimas ferramentas que permitem ao cliente visualizar exatamente a ação do sistema e, conseqüentemente, aprovar ou não, avaliando se está de acordo com o que foi solicitado e se atende às regras de negócios especificadas.


Veja no Quadro 10 os principais elementos componentes de um caso de uso:

Quadro 10 – Componentes do caso de uso

Termo e definição	Símbolo
<p>Um ator</p> <ul style="list-style-type: none"><li>• É uma pessoa ou um sistema que obtém benefícios do sistema e é externo a ele.</li><li>• É identificado por sua função.</li><li>• Pode ser associado a outros por uma associação de especialização/superclasse.</li><li>• É colocado no lado externo da fronteira do sistema.</li></ul>	

Termo e definição	Símbolo
<ul style="list-style-type: none"> <li>• Um caso de uso.</li> <li>• Representa uma parte importante da funcionalidade do sistema.</li> <li>• Pode ampliar outro caso de uso.</li> <li>• Pode empregar outro caso de uso.</li> <li>• É colocado no lado interno da fronteira do sistema.</li> <li>• É identificado com uma expressão descritiva verbo- substantivo.</li> </ul>	
<ul style="list-style-type: none"> <li>• Uma fronteira do sistema.</li> <li>• Inclui o nome do sistema em seu interior ou na sua parte superior.</li> <li>• Representa o escopo do sistema.</li> </ul>	
<ul style="list-style-type: none"> <li>• Um relacionamento de associação.</li> <li>• Liga um ator ao(s) caso(s) de uso com o(s) qual(is) ele interage.</li> </ul>	

Fonte: DENNIS, 2014, p. 504.



Conforme o quadro evidencia, os atores são representados por bonecos de palito e podem exemplificar as ações de outros sistemas ou de usuários, ou seja, uma entidade externa. O ator basicamente representa a ação que um usuário poderá executar em um sistema, chamada de função (DENNIS, 2014).

É preciso saber identificar quais serão os atores do sistema. Então, quem utilizará o sistema de uma forma geral, tanto no que diz respeito a pessoas quanto a outros sistemas de informação deverão ser considerados.

## Para saber mais

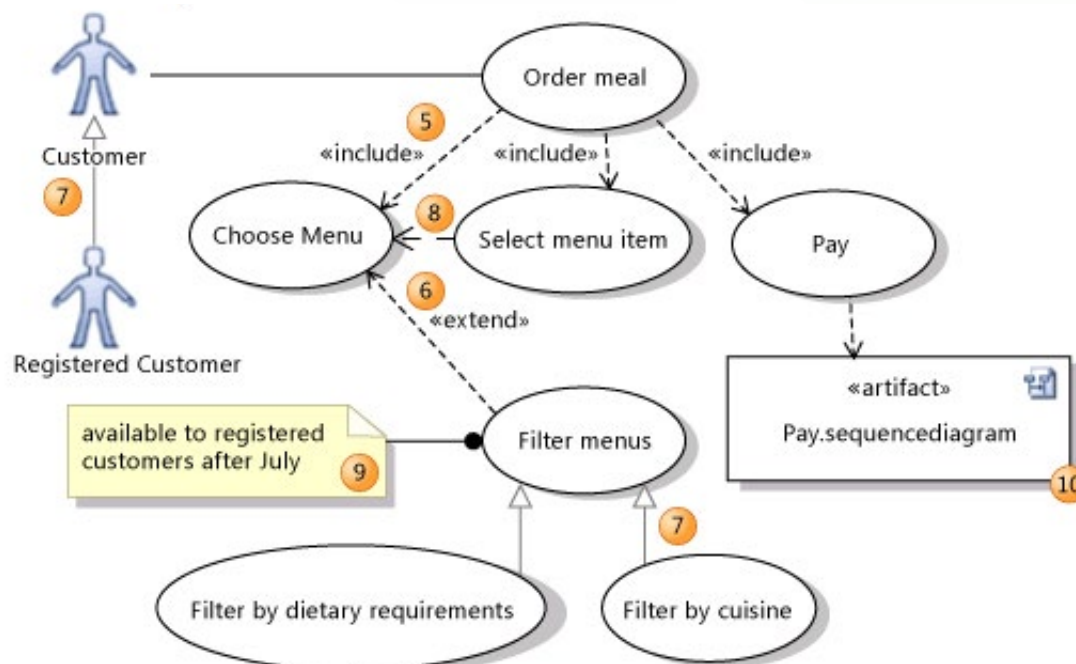
Leia o artigo UML Use Case Diagrams: Guidelines. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409427.aspx>>. Acesso em: 17 maio 2016.

Um ator também pode ser um *especialista*, ou seja, que desempenhará uma função no sistema de forma diferenciada. Será representado por uma linha com um triângulo sem preenchimento, vazado, e que aponta diretamente para uma superclasse.

Um exemplo é se temos um módulo do sistema para cadastro das pessoas que poderão utilizar o sistema, e criamos outras subclasses clientes, fornecedores,

vendedores. Todos são atores, porém cada um exercerá uma função distinta naquele determinado módulo do sistema. Dessa forma, está correto dizer que haverá a representação de uma herança. Um ator especialista herdará todas as características, ou seja, os atributos fruto dessa **generalização**. A seguir uma figura e um quadro para exemplificar esses conceitos:

Figura 15 – Estruturação de um estudo de caso



Fonte: MSDN, Microsoft. Diagramas de caso de uso UML: referência. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409427.aspx>>. Acesso em: 17 maio 2016.

Agora no Quadro 11, disponível também pelo MSDN Microsoft, estão explicadas cada uma das interações que ocorrem na figura 16. Analise a seguir:

Quadro 11 – Elementos de interação

Forma	Elemento	Descrição
5	Incluir (<<include>>)	Um caso de uso de inclusão. É usado para mostrar como dividir um caso de uso em etapas menores. Observe que o diagrama não mostra a ordem das etapas. Você pode usar um diagrama de atividades de sequência para fazer esse detalhamento.
6	Estender <<extend>>	Um caso de uso estendido adiciona ou compartilha os métodos com outro caso de uso. As extensões operam apenas sob determinadas condições. Pode usar um comentário para estabelecer a condição.
7	Herança	Esse tipo de representação significa que o ator ou caso de uso carrega as mesmas características que o indicado pelo fluxo (seta vazada). Herda os métodos e atributos do ator ou casos de uso que foram envolvidos na generalização.
8	Dependência	Indica uma relação de interfaces em que uma dependerá da outra para ser acionada pelo sistema.

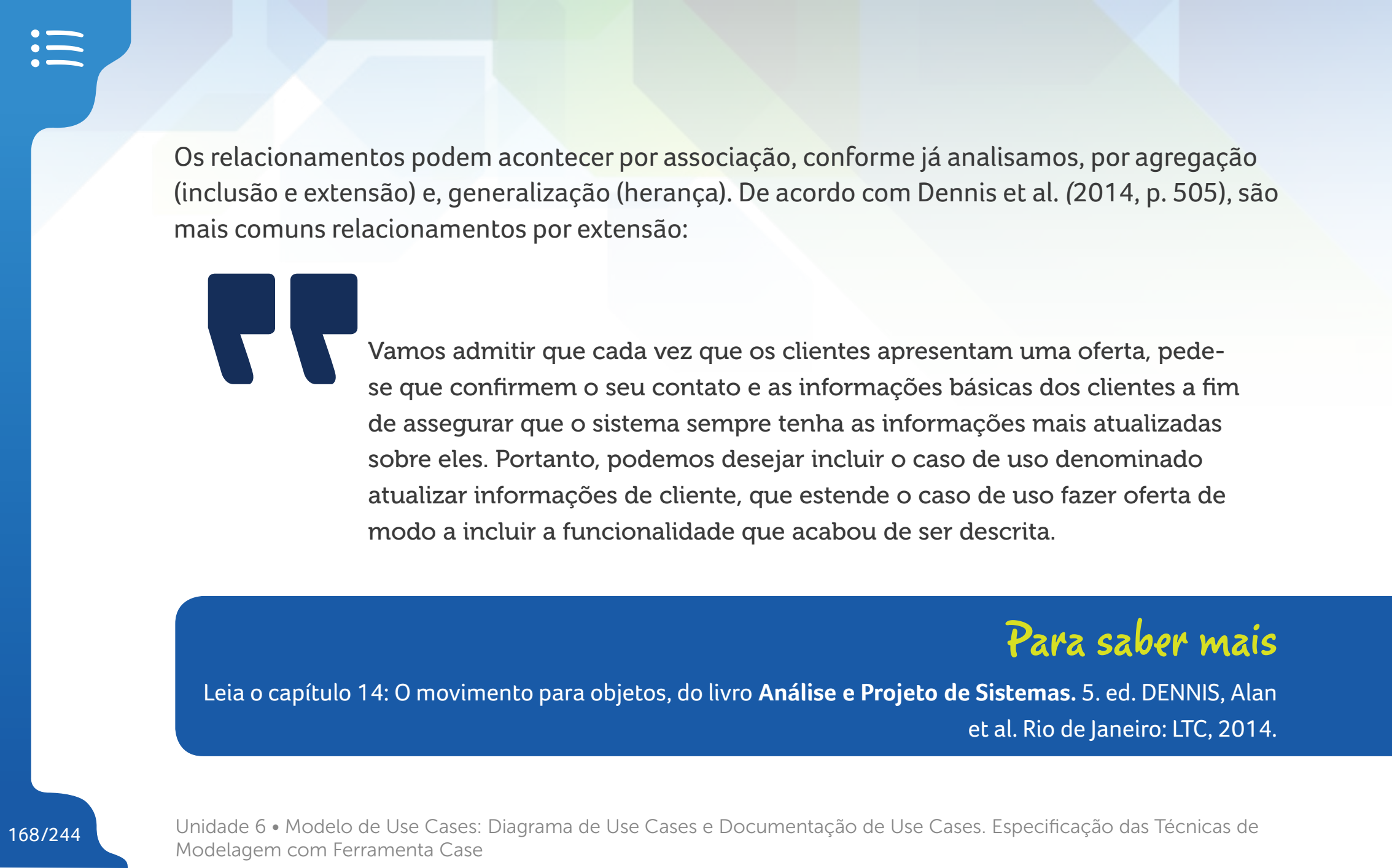
Forma	Elemento	Descrição
9	Comentário	Usado para adicionar informações ao sistema.
10	Artefato	Esse fornece um link ou hiperlink para outro documento. Um artefato pode ser utilizado para vincular um caso de uso a um diagrama de sequência por exemplo.

Fonte: Adaptado de MSDN, Microsoft. Diagramas de caso de uso UML: referência. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409427.aspx>>. Acesso em: 17 maio 2016.

Um caso de uso pode incluir ou estender algumas funcionalidades de um outro caso de uso; é o que se obtém dos **relacionamentos**, conforme visualizado na seção anterior deste material.



Leia mais sobre os casos de uso em: <<http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/usecases.htm>>. Acesso em: 17 maio 2016.



Os relacionamentos podem acontecer por associação, conforme já analisamos, por agregação (inclusão e extensão) e, generalização (herança). De acordo com Dennis et al. (2014, p. 505), são mais comuns relacionamentos por extensão:



Vamos admitir que cada vez que os clientes apresentam uma oferta, pede-se que confirmem o seu contato e as informações básicas dos clientes a fim de assegurar que o sistema sempre tenha as informações mais atualizadas sobre eles. Portanto, podemos desejar incluir o caso de uso denominado atualizar informações de cliente, que estende o caso de uso fazer oferta de modo a incluir a funcionalidade que acabou de ser descrita.

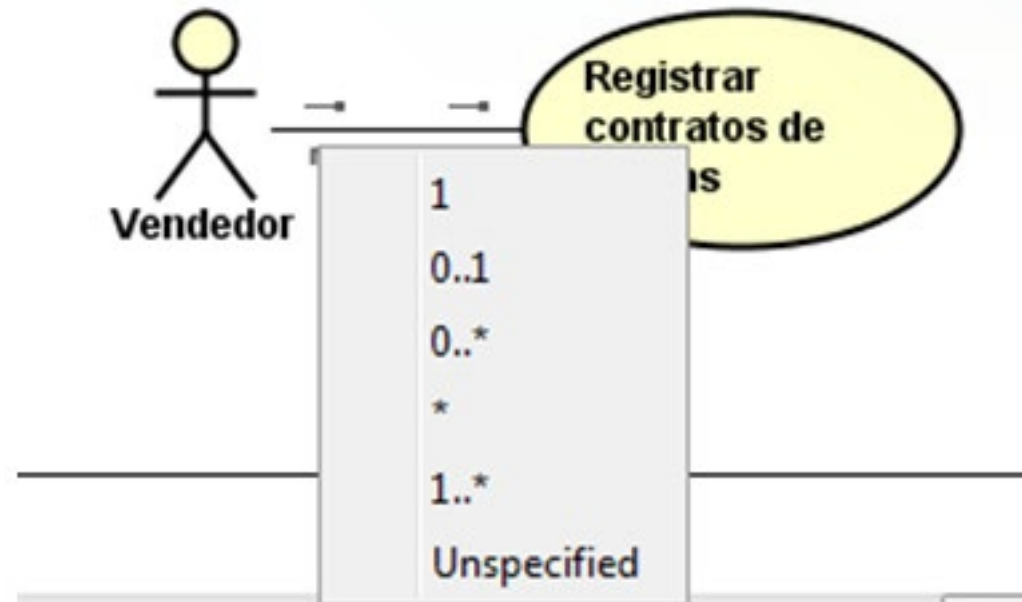
## *Para saber mais*

Leia o capítulo 14: O movimento para objetos, do livro **Análise e Projeto de Sistemas**. 5. ed. DENNIS, Alan et al. Rio de Janeiro: LTC, 2014.

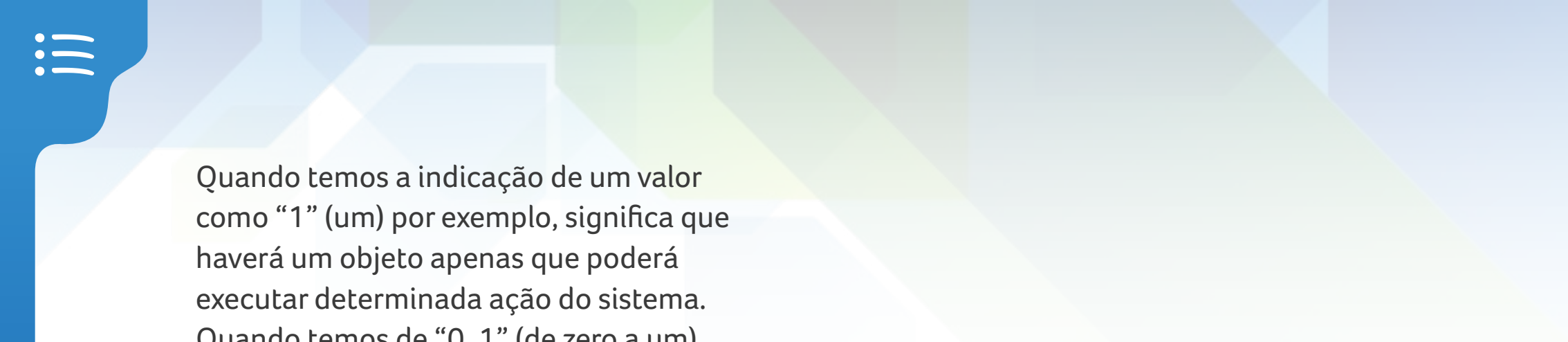


Outro fator importante a considerar é a questão da **multiplicidade**, que pode indicar a quantidade de objetos interações que o objeto contém, ou seja, a quantidade de vezes que um objeto (instância) pode executar a ação do sistema, nesse caso representada pelo caso de uso:

Figura 16 – Tipos de multiplicidade



Fonte: Elaborada pelo autor.



Quando temos a indicação de um valor como “1” (um) por exemplo, significa que haverá um objeto apenas que poderá executar determinada ação do sistema. Quando temos de “0..1” (de zero a um) uma faixa de valores, isso denota que de zero a “n” ou a “1” objetos que poderão utilizar os métodos do caso de uso. Se a especificação estiver com “0..\*” (zero a muitos), representa que de zero a muitos objetos podem utilizar aquela ação. Se tiver apenas o símbolo do asterisco “\*” (muitos para muitos, ou, multiplicidade), significa que muitos atores (objetos) poderão utilizar o caso de uso, diz-se de muitos para muitos. Se a representação for “1..\*” (um para muitos) objetos poderão interagir.

## Para saber mais

Assista ao vídeo sobre diagramas de casos de uso e o que significa. Disponível em: <<https://www.youtube.com/watch?v=JhnVqARE39U>>.

Acesso em: 17 maio 2016.



## Glossário

**Incidência:** quantidade de ocorrências de um determinado fato ou contexto.

**Multiplicidade:** em modelagem de dados, refere-se ao conjunto de elementos contidos em um conjunto, ou determinação da quantidade de objetos que podem conter as classes, atores e os casos de uso.

**Diagramas:** gráficos que representam entidades, casos de uso, atores e suas relações em modelagem de dados.



# Questão para reflexão

Como criar diagramas de casos de uso que atendam e representem as funcionalidades do sistema e, ainda, permitam a inclusão desde o projeto de novos incrementos?





## Considerações Finais

Neste tema, pudemos conhecer o que são casos de uso e o que é um diagrama de caso de uso. Você pôde lembrar ou conhecer quais são os principais tipos de associação que podem ser realizadas em casos de uso: associação, agregação ou generalização. Além disso, compreendeu que a cardinalidade estabelece a quantidade de objetos com que um ator ou caso de uso podem interagir e, ainda, pôde analisar as formas de se identificar a necessidade de criar um caso de uso.



## Referências

DENNIS, Alan et al. **Análise e Projeto de Sistemas**. 5. ed. Rio de Janeiro: LTC, 2014.

MACORATTI, José Carlos. Modelando sistemas em UML: casos de uso. Disponível em: <[http://www.macoratti.net/net\\_uml2.htm](http://www.macoratti.net/net_uml2.htm)>. Acesso em: 16 maio 2016.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

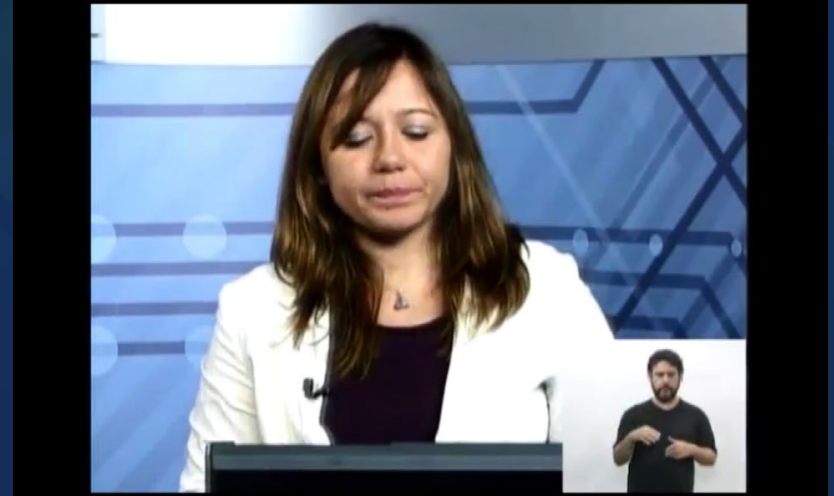


# Assista a suas aulas



## **Aula 6 - Tema: Modelo de use Cases: Diagrama de use Cases e Documentação de use Cases - Bloco I**

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/debb1eb92b2cec632939b0f2fabb9962>>.



## **Aula 6 - Tema: Modelo de use Cases: Diagrama de use Cases e Documentação de use Cases - Bloco II**

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/93d2348b67c564f81b1dca8454a09eda>>.



# Questão 1

## 1. Assinale a alternativa correspondente:

- I – Com os casos de uso, é possível descrever as interações que ocorrerão no sistema, ou seja, ações entre usuários e sistema, bem como a interação entre sistemas.
- II – O modo de representação de um caso de uso é bem simples.
- III – Considera a forma geométrica elipse, para representar a ação, e atores, para indicar quem envia e quem recebe a solicitação.

São verdadeiras:

- a) I e II apenas.
- b) I e III apenas.
- c) I, II e III.
- d) Apenas a alternativa III.
- e) III e II.





## Questão 2

### 2. Complete a frase com termos essenciais e, respectivamente, dispostos nas alternativas:

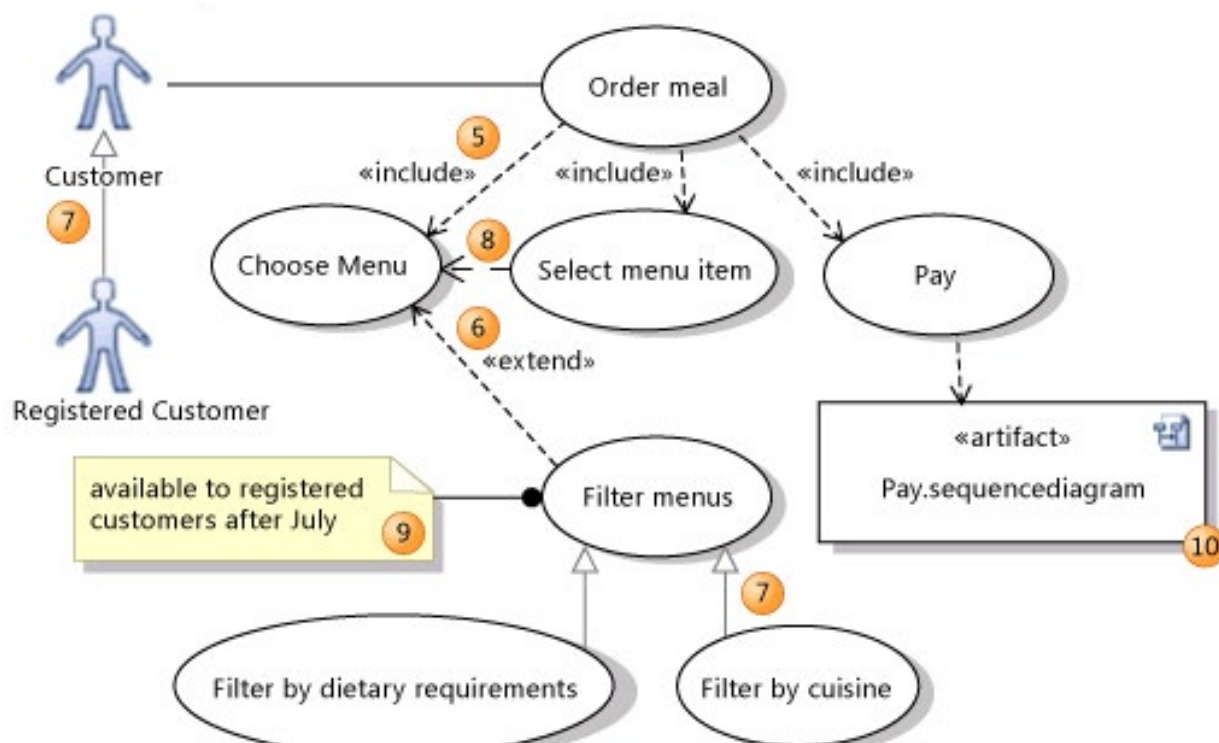
“\_\_\_\_\_ dão uma visão simples de uma interação. Logo, é necessário fornecer mais detalhes para entender o que está envolvido. Esses detalhes podem ser uma \_\_\_\_\_, uma descrição estruturada em uma tabela ou um diagrama de sequência [...]. Você deve escolher o formato mais adequado, dependendo do caso de uso, e o nível de detalhamento que você acredita ser necessário no modelo” (SOMMERVILLE, 2011, p. 87).

- a) Diagramas de caso de uso / simples descrição textual
- b) Diagramas de sequência / português estruturado
- c) Diagramas de caso de uso / diagramas de fluxo de dados
- d) Estórias / diagramas de caso de uso
- e) Multiplicidade / diagramas de caso de uso

# Questão 3

3.

Figura 17 – Estruturação de um estudo de caso



Fonte: MSDN, Microsoft. Diagramas de caso de uso UML: referência. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409427.aspx>>. Acesso em: 17 maio 2016.



## Questão 3

I – O item “5” representa uma associação por generalização ou herança.

II – O item “7” representa uma associação por generalização ou herança.

III – O item “9” representa um comentário.

São verdadeiras:

- a) I, II e III.
- b) I e II.
- c) I e III.
- d) II e III.
- e) III apenas.



## Questão 4

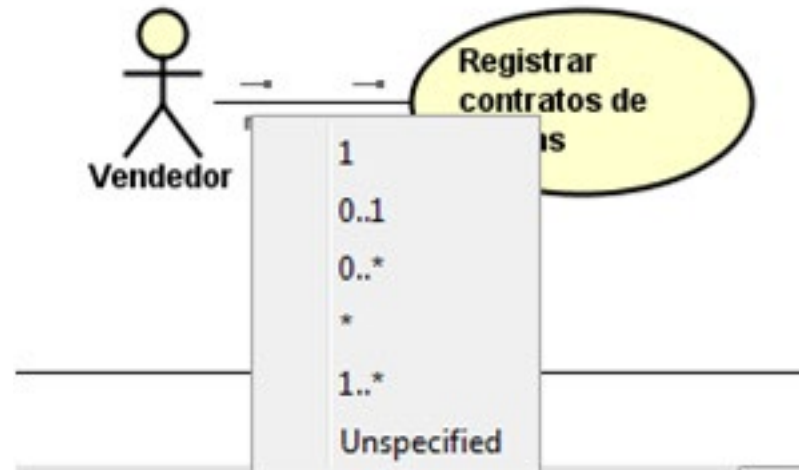
Os elementos do quadro referem-se, respectivamente, a:

Elemento	Descrição
I.	É usado para mostrar como dividir um caso de uso em etapas menores. Observe que o diagrama não mostra a ordem das etapas. Você pode usar um diagrama de atividades de sequência para fazer este detalhamento.
II.	Um caso de uso estendido adiciona ou compartilha os métodos com outro caso de uso. As extensões operam apenas sob determinadas condições. Pode usar um comentário para estabelecer a condição.
III.	Indica uma relação de interfaces em que uma dependerá da outra para ser acionada pelo sistema.

- a) I – Dependência; II – Estender; III – Incluir.
- b) I – Incluir; II – Estender; III – Dependência.
- c) I – Estender; II – Incluir; III – Dependência.
- d) I – Dependência; II – Incluir; III – Estender.
- e) I – Exclusão; II – Inclusão; III – Extensão.

## Questão 5

5. A figura abaixo refere-se principalmente à:



- a) seleção de um caso de uso;
- b) determinação de ator;
- c) determinação de ação;
- d) seleção de subsistema em UML;
- e) determinação de multiplicidade.

## 1. Resposta: C.

Comentário: observe que fica mais fácil compreender a ação do sistema, e como está interagindo com o ambiente externo.

Figura 18 – Exemplo de caso de uso em um sistema de cadastro



Fonte: Adaptado de Sommerville, 2011, p. 86.

## 2. Resposta: A.

Comentário: as descrições podem ser melhor aproveitadas para aumentar a produtividade da equipe e maximizar a qualidade do produto de software. Assim, as teorias se convergem e a prática ganha em resultados.



## 3. Resposta: D.

Comentário:

Forma	Elemento	Descrição
5	Incluir (<<include>>)	Um caso de uso de inclusão. É usado para mostrar como dividir um caso de uso em etapas menores. Observe que o diagrama não mostra a ordem das etapas. Você pode usar um diagrama de atividades de sequência para fazer este detalhamento.
7	Herança	Esse tipo de representação significa que o ator ou caso de uso carrega as mesmas características que o indicado pelo fluxo (seta vazada). Herda os métodos e atributos do ator ou casos de uso que foram envolvidos na generalização.
9	Comentário	Usado para adicionar informações ao sistema.



## 4. Resposta: B.

Comentário:

Elemento	Descrição
I – Incluir	É usado para mostrar como dividir um caso de uso em etapas menores. Observe que o diagrama não mostra a ordem das etapas. Você pode usar um diagrama de atividades de sequência para fazer este detalhamento.
II – Estender	Um caso de uso estendido adiciona ou compartilha os métodos com outro caso de uso. As extensões operam apenas sob determinadas condições. Pode usar um comentário para estabelecer a condição.
III – Dependência	Indica uma relação de interfaces em que uma dependerá da outra para ser acionada pelo sistema.

## 5. Resposta: E.

Comentário: Quando temos a indicação de um valor como “1” (um), por exemplo, significa que haverá um objeto apenas que poderá executar determinada ação do sistema. Quando temos de “0..1” (de zero a um) uma faixa de valores, isso denota que de zero a “n” ou a “1” objetos





# Gabarito

poderão utilizar os métodos do caso de uso. Se a especificação estiver com “0..\*” (zero a muitos), representa que de zero a muitos objetos podem utilizar aquela ação. Se tiver apenas o símbolo do asterisco “\*” (muitos para muitos, ou, multiplicidade), significa que muitos atores (objetos) poderão utilizar o caso de uso; diz-se de muitos para muitos. Se a representação for “1..\*” (um para muitos) objetos poderão interagir.



# Unidade 7

## Diagrama de Classes e de Sequência

---

### Objetivos

Com esta disciplina, temos os objetivos de:

1. 1. Apresentar onde e como os diagramas de classe e de sequência são utilizados para a análise de sistemas.
2. 2. Conhecer como os diagramas de classe e de sequência podem representar a estrutura e o comportamento do sistema.
3. 3. Incentivar a análise e a avaliação de modelos de projetos e como podem servir de exemplos a contratos que venham a se engajar.



## Introdução

Primeiramente, será apresentada neste tema a técnica para o desenvolvimento de um diagrama de classes, sendo esse considerado estático, pois visa definir a estrutura do sistema. Auxilia de modo eficaz sua visualização e principais relacionamentos, métodos, comportamentos e estados (DENNIS et al., 2014).

Na segunda seção deste material, você conhecerá a técnica de desenvolvimento de um diagrama de sequência. Esse visa representar todos os objetos contidos em um caso de uso, e a forma como acontece a comunicação ou interação entre eles, com a passagem das mensagens. Ele é considerado **dinâmico**, pois apresenta como acontecerá a sequência dos eventos

do sistema de forma sequencial, como o próprio nome diz, e ordenada, portanto.


Como estudado, o elemento-base que compõe o paradigma orientado a objetos e que é comum aqui é a classe, portanto, a responsabilidade da classe é armazenar e gerenciar as informações de um dado sistema. Observe o Quadro 12 abaixo, que contém as respectivas definições dos elementos de um diagrama de classes:

Quadro 12 – Descrição dos elementos de uma classe

Termo e Definição	Símbolo
<b>Uma classe</b> <ul style="list-style-type: none"><li>• Representa um tipo de pessoa, um lugar ou algo sobre o qual o sistema deve capturar e armazenar informações.</li><li>• Tem o nome em negrito e centrado no seu comportamento superior.</li><li>• Tem uma lista de atributos no seu compartimento inferior.</li><li>• Não mostra explicitamente as operações que estão disponíveis para todas as classes.</li></ul>	<b>Nome da classe</b>  - Nome do atributo  -/ nome do atributo derivado  + Nome da operação ()
<b>Um atributo</b> <ul style="list-style-type: none"><li>• Representa as propriedades que descrevem o estado de um objeto.</li><li>• Pode ser derivado de outros atributos, o que é mostrado pela colocação de uma barra inclinada antes do nome do atributo.</li></ul>	Nome do atributo/ nome do atributo derivado

Termo e Definição	Símbolo
<p><b>Um método</b></p> <ul style="list-style-type: none"> <li>• Representa as ações ou funções que uma classe pode realizar.</li> <li>• Pode ser classificado como uma operação construtora, de consulta ou de atualização.</li> <li>• Inclui parênteses que podem conter parâmetros especiais ou informações necessárias para que a operação seja realizada.</li> </ul> <p><b>Uma associação</b></p> <ul style="list-style-type: none"> <li>• Representa um relacionamento entre várias classes, ou de uma classe consigo.</li> <li>• É identificada por uma expressão verbal ou pelo nome de uma função, o que representar melhor o relacionamento.</li> <li>• Pode existir entre uma ou mais classes.</li> <li>• Contém símbolos de multiplicidade, que representam o número mínimo e o número máximo de vezes que uma instância pode ser associada à instância da classe relacionada.</li> </ul>	<p>Nome da operação ()</p> <p>1..* expressão verbal 0..1</p>

Fonte: DENNIS et.al, 2014, p. 509.



No processo de análise do sistema, as classes sempre representarão as formas como o sistema armazenará as informações, ou seja, pode ser uma classe “Pessoa”, ou mesmo “Local”, por exemplo. Já na fase de implementação, uma classe pode representar um formulário ou uma outra ação do sistema. Como visto no Quadro 12, a representação de uma classe se dá por meio de um retângulo, dividido em três linhas que contém respectivamente, o nome da classe, os seus atributos e, por fim, os métodos da classe.


Tanto os atributos quanto os métodos precisam da inserção da sinalização de visibilidade: + pública, - privada, # protegido, o que quer dizer que nem todos os demais objetos ou mesmo outras classes poderão acessar um determinado atributo ou método.

Os métodos de uma classe indicam o seu comportamento, ou a ação que a classe executará ao ser solicitada uma execução. As ações do sistema podem ser: um método construtor como `inserir()`; um método de consulta como `somar()`; um método de atualização como `atualizar()`, entre outras ações que se pode criar, de acordo com as regras de negócios que se pretende atender (DENNIS et al., 2014, p. 510):



O método construtor cria uma nova instância de uma classe. Por exemplo a classe veículo pode ter um método chamado `inserir()` que crie uma nova instância dessa classe. [...] Um método de consulta (query) gera informações sobre o estado de um objeto disponível para outros objetos, mas não alterará o objeto de forma alguma. [...] admitimos que todos os objetos possuem métodos (ou operações) que produzem os valores de seus atributos. [...] Um método de atualização alterará o valor de algum ou de todos os atributos do objeto, o que pode resultar em uma alteração no estado do objeto.

Dessa forma, é possível orientar as operações que uma determinada classe terá, e também a forma como estabelecerá as associações com outras classes. Uma associação assemelha-se ao uso de ponteiros, pois indica exatamente o ponto que se deseja estabelecer a comunicação, ou troca de informações. As associações também podem ser por generalização, em que uma subclasse herda as características ou atributos e os métodos da superclasse (representadas pela seta vazada). E também podem se associar por agregação, com a inclusão de novas classes ou instâncias; isso significa que englobam, literalmente, outras classes (é representada



pelo fluxo seguido de um losango. São facilmente identificadas com os termos “faz parte de”, “é composto de”) (DENNIS et al., 2014, p. 512).

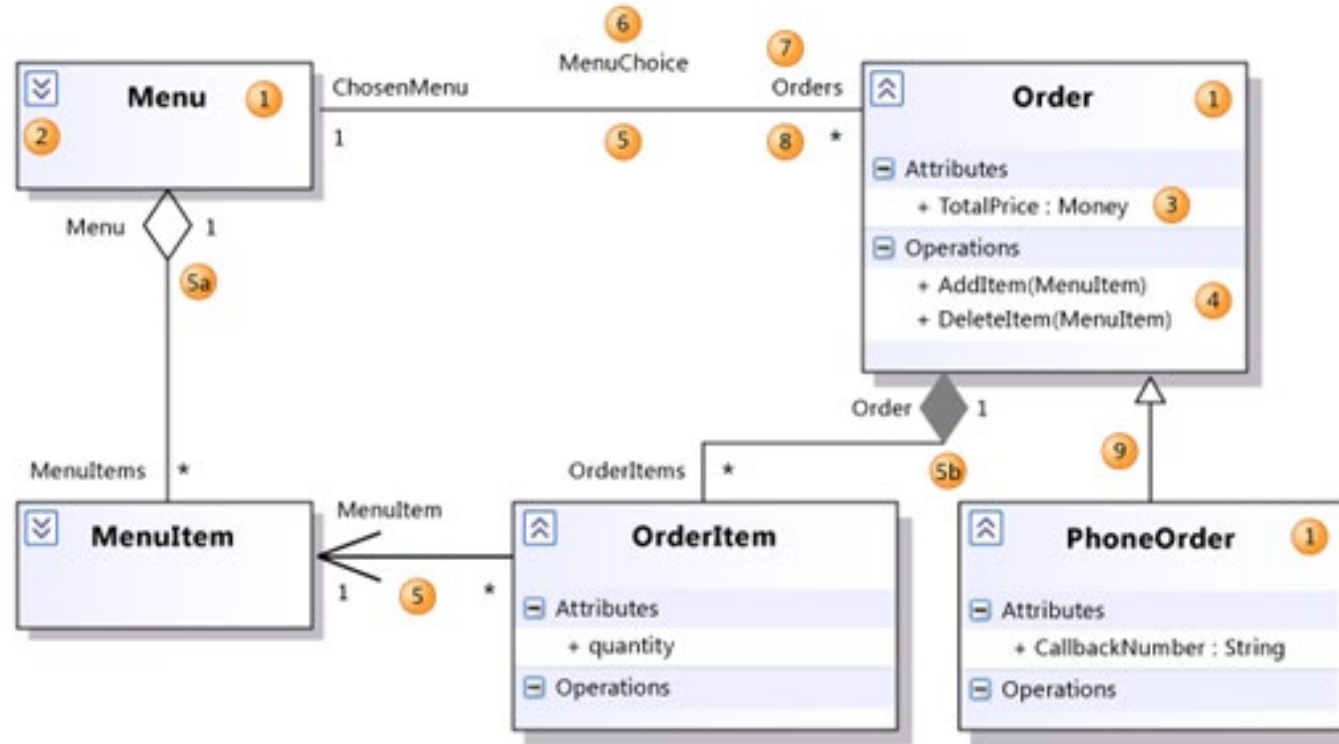
Observe a figura a seguir e na sequência o quadro que descreve cada um de seus elementos, de forma a exemplificar os assuntos tratados até o momento:



Aprenda mais com a leitura do artigo sobre Diagramas de classe: referência. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409437.aspx>>. Acesso em: 18 maio 2016.



Figura 19 – Diagrama de classes



Fonte: MSDN , Microsoft. Diagramas de classe UML: referência. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409437.aspx>>. Acesso em: 18 maio 2016.

Observe que o item de número 1 representa a **classe**, assim compartilha objetos comportamentais e estruturais e também a sua representa a sua classificação, por isso também



pode ser chamada de **classificador**. Atores e casos de uso também são classificadores. O elemento 2 nada mais é do que um item que permite recolher ou **expandir** o controle para visualizar detalhes da classificação (+).

O item de número 3 é um **atributo** da classe Order, com visibilidade pública, ou seja, pode ser acessado por outros objetos. Já o item 4 desse diagrama representa uma operação, ou **método**. No caso, na representação do método 4, temos um que é AddItem(MenuItem) e o outro que é DeletarItem (MenuItem), ambos com **visibilidade** pública (+) e pertencentes à classe MenuItem, compartilhados com a classe Order por meio de agregação por

composição (representada pelo losango preenchido). O item 5 representa uma associação entre as classes indicadas. Daí, surgem duas novas associações, a 5a, que é por **agregação**, e a 5b, que é por **composição**.

O elemento 6 representa o nome da associação. Já o elemento 7, representa, respectivamente o nome da função, muito utilizado para referenciar um objeto, apontar a ele. Uma função pode ter suas próprias características e propriedades.

Elemento de número 8 indica a questão da multiplicidade, já estudada no tema anterior. O item 9 representa uma **generalização** ou herança.



## Para saber mais

Leia o artigo “Utilizando UML: diagrama de classes”. Disponível em: <<http://www.devmedia.com.br/artigo-sql-magazine-63-utilizando-uml-diagrama-de-classes/12251>>. Acesso em: 18 maio 2016.

## Para saber mais

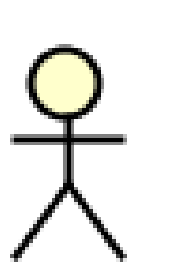


Vídeo que mostra outra ferramenta, o Star UML e como desenvolver um diagrama de sequência. Disponível em: <<https://www.youtube.com/watch?v=UKfdmCMVda4>>. Acesso em: 19 maio 2016.

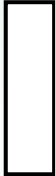


### 1. Diagramas de Sequência

Os diagramas de sequência podem ser **genéricos**, em que é possível demonstrar todos os **cenários** de um caso de uso. E também podem ser **diagramas de sequência de instâncias**, que representam apenas um cenário de um caso de uso (DENNIS et al., 2014).

A seguir, no Quadro 13, os elementos de um diagrama de sequência:

Quadro 13 – Elementos de um diagrama de sequência

Termo e Definição	Símbolo
<b>Um ator</b> <ul style="list-style-type: none"><li>• É uma pessoa ou um sistema que obtém benefícios do sistema e é externo a ele.</li><li>• Participa em uma sequência enviando e/ou recebendo mensagens.</li><li>• É colocado ao longo da parte superior do diagrama.</li></ul>	
<b>Um objeto</b> <ul style="list-style-type: none"><li>• Participa em uma sequência enviando e/ou recebendo mensagens.</li><li>• É colocado ao longo da parte superior do diagrama.</li></ul>	
<b>Uma linha de vida</b> <ul style="list-style-type: none"><li>• Indica a vida de um objeto durante uma sequência.</li><li>• Contém um X no ponto em que a classe deixa de interagir</li></ul>	

Termo e Definição	Símbolo
<b>Um foco de controle</b> <ul style="list-style-type: none"> <li>É um retângulo longo e estreito colocado sobre uma linha de vida.</li> <li>Indica quando um objeto está enviando ou recebendo mensagens.</li> </ul>	
<b>Uma Mensagem</b> <ul style="list-style-type: none"> <li>Transmite informações de um objeto para o outro.</li> </ul>	
<b>Destruição de um objeto</b> <ul style="list-style-type: none"> <li>É colocado um X no final da linha de vida de um objeto para mostrar que ele está deixando de existir.</li> </ul>	

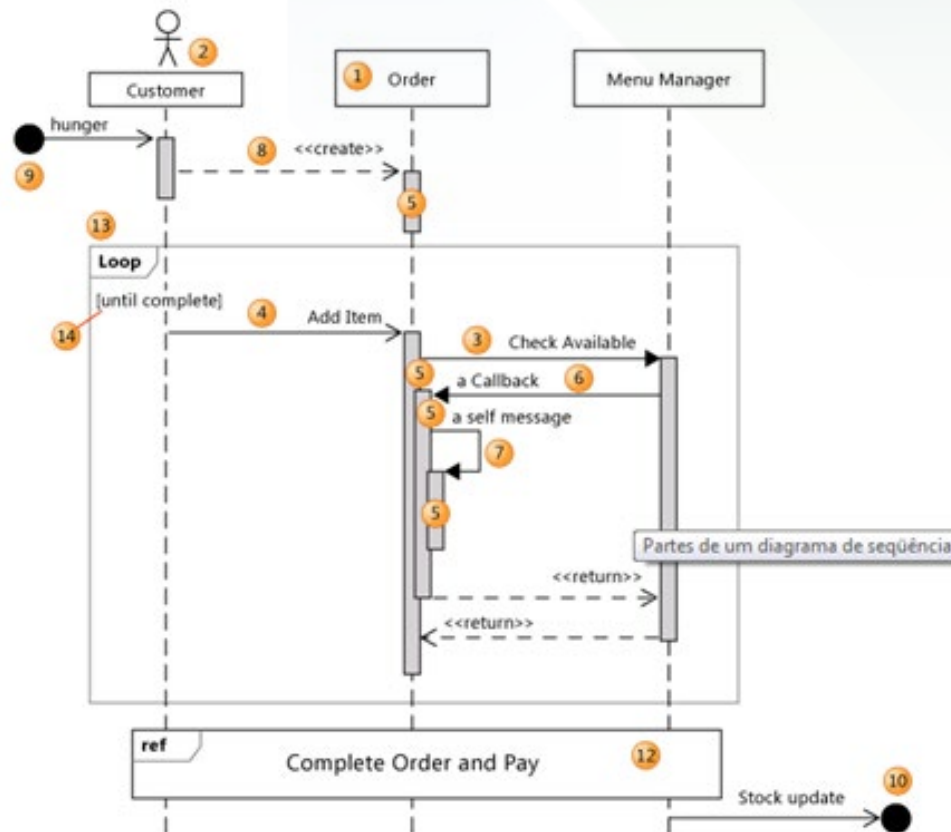
Fonte: DENNIS et al., 2014, p. 517.



Aprenda mais com a leitura do artigo “Diagramas de sequência UML: referência”. Disponível em:  
 <<https://msdn.microsoft.com/pt-br/library/dd409377.aspx>>. Acesso em: 19 maio 2016.

Agora, vamos conhecer como é um diagrama de sequência:

Figura 20 – Diagrama de sequência



Fonte: MSDN, Microsoft. Artigo Diagramas de sequência UML: referência. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409377.aspx>>. Acesso em: 20 maio 2016.

## Para saber mais

Acesse os conteúdos da Universia em: <[http://biblioteca.universia.net/html\\_bura/ficha/params/title/transforma%C3%A7%C3%A3o-modelos-diagrama-sequ%C3%Aancia-uml-contemplando-restri%C3%A7%C3%B5es-tempo-energia-rede/id/56750378.html](http://biblioteca.universia.net/html_bura/ficha/params/title/transforma%C3%A7%C3%A3o-modelos-diagrama-sequ%C3%Aancia-uml-contemplando-restri%C3%A7%C3%B5es-tempo-energia-rede/id/56750378.html)>.

Acesso em: 19 maio 2016.

Vamos às respectivas descrições dos elementos da Figura 20:

1. Linha da vida: representando a sequência de mensagens trocadas por um objeto durante a interação.
2. Ator: pode ser uma classe, um caso de uso.
3. Mensagem síncrona: O remetente

recebe a resposta do sistema antes de continuar a transmissão e enviar a mensagem.

4. Mensagem assíncrona: não requer resposta.
5. Ocorrência de execução: representa o período que o objeto leva para executar a ação.
6. Mensagem de retorno de chamada: retorno de uma chamada anterior ao solicitante.
7. Mensagem self: uma mensagem do próprio objeto.
8. Criar mensagem: gera mensagem a outro objeto e é o primeiro a enviar.

- 9. Mensagem encontrada: uma mensagem desconhecida ou de um comportamento desconhecido ou externo ao programa.
- 10. Mensagem perdida: ocorre quando não há a especificação de um objeto de envio.
- 11. Comentário: pode ser inserido em qualquer ponto do diagrama.
- 12. Uso de interação: prevê a interação em um determinado período com outro sistema ou objeto.
- 13. **Fragmento** combinado: inclui uma ou mais mensagens. Pode estar associado a uma função, como uma estrutura de repetição.

- 14. Protetor de fragmento: determina uma condição ao fragmento combinado.

Fonte: Adaptado de <<https://msdn.microsoft.com/pt-br/library/dd409377.aspx>>. Acesso em: 20 maio 2016.



Estude mais em: <[http://www.macoratti.net/net\\_uml3.htm](http://www.macoratti.net/net_uml3.htm)>. Acesso em: 19 maio 2016.





## Glossário

**Cenários:** caminho executável e único através de um caso de uso (DENNIS, 2014, p. 515).

**Dinâmico:** que se modifica, que pode evoluir ou mudar de comportamento.

**Fragmento:** significa que contempla apenas parte do todo, ou apenas uma ação do sistema.



# Questão para reflexão

Quais são as características do diagrama de sequência que o torna um dos quatro fundamentais da UML? Pense nisso!





## Considerações Finais

Primeiramente, com uma revisão dos principais conceitos pertinentes ao desenvolvimento do diagrama de classes, pudemos observar a sua composição e estrutura.

Em seguida, ao observar a Figura 1, foi possível perceber quantas interações e ações são necessárias no diagrama de classes e, a importância de se estabelecer corretamente as associações.

Além desses ensinamentos, com os estudos sobre os diagramas de sequência, foi possível retomar a ideia de que podem ser desenvolvidos de forma generalista, ou genérica, com a visão do cenário de um caso de uso, e também que podem ser elaborados sob um ponto de vista mais específico com base em uma instância, denotando o seu comportamento.

Para finalizar, você pode compreender de que forma interagem outros elementos em um diagrama de sequência e de que forma as mensagens são consideradas por um sistema e podem ser representadas.



## Referências

DENNIS, Alan et al. **Análise e Projeto de Sistemas**. 5. ed. Rio de Janeiro: LTC, 2014.

MACORATTI, José Carlos. Modelando sistemas em UML: casos de uso. Disponível em: <[http://www.macoratti.net/net\\_uml2.htm](http://www.macoratti.net/net_uml2.htm)>. Acesso em: 16 maio 2016.



# Assista a suas aulas



## Aula 7 - Tema: Diagrama de Classes e de Sequência - Bloco I

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/1cb3d43940ef5aab2f24828bc39d92bd>>.



## Aula 7 - Tema: Diagrama de Classes e de Sequência - Bloco II

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/c8141d9a953526742ca67a01b9e5dc78>>.



# Questão 1

## 1. Os aspectos abaixo referem-se a que elemento?

I – Representa um tipo de pessoa, um lugar ou algo sobre o qual o sistema deve capturar e armazenar informações.

II – Tem o nome em negrito e centrado no seu comportamento superior.

III – Tem uma lista de atributos no seu compartimento inferior.

IV – Não mostra explicitamente as operações que estão disponíveis para todas as classes.

- a) Atributo
- b) Classe
- c) Método
- d) Visibilidade
- e) Multiplicidade



## Questão 2

### 2. A sequência abaixo mostra, respectivamente, informações sobre:

I – Representa as ações ou funções que uma classe pode realizar.

II – Pode ser classificado como uma operação construtora, de consulta ou de atualização.

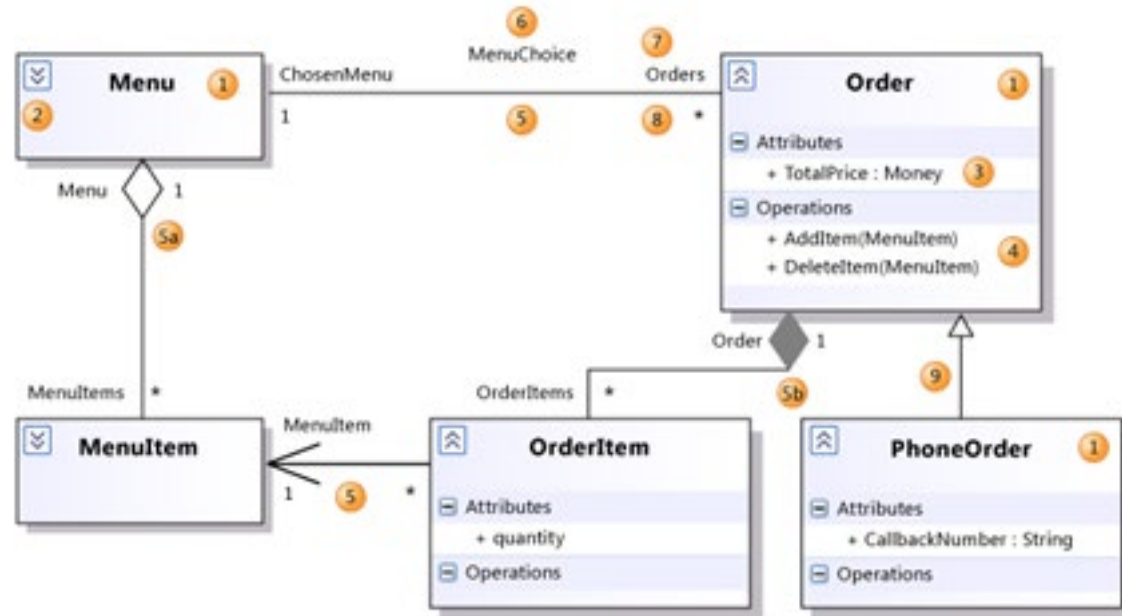
III – Inclui parênteses que podem conter parâmetros especiais ou informações necessárias para que a operação seja realizada.

- a) Métodos
- b) Atributos
- c) Classes
- d) Agregações
- e) Composições



## Questão 3

3. Assinale a alternativa que traz o nome do diagrama da figura a seguir:



Fonte: MSDN , Microsoft. Diagramas de classe UML: referência. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409437.aspx>>. Acesso em: 18 maio 2016.





## Questão 3

- a) Diagrama de sequência.
- b) Diagrama de caso de uso.
- c) Diagrama de estado.
- d) Diagrama de classe.
- e) Diagrama de atividades.

# Questão 4

## 4. Os itens do quadro abaixo representam, respectivamente:

Termo e Definição	Símbolo
<ul style="list-style-type: none"> <li>Participa em uma sequência enviando e/ou recebendo mensagens.</li> <li>É colocado ao longo da parte superior do diagrama.</li> </ul>	
<ul style="list-style-type: none"> <li>Indica a vida de um objeto durante uma sequência.</li> <li>Contém um X no ponto em que a classe deixa de interagir.</li> </ul>	
<ul style="list-style-type: none"> <li>É um retângulo longo e estreito colocado sobre uma linha de vida.</li> <li>Indica quando um objeto está enviando ou recebendo mensagens.</li> </ul>	
<ul style="list-style-type: none"> <li>Transmite informações de um objeto para o outro.</li> </ul>	

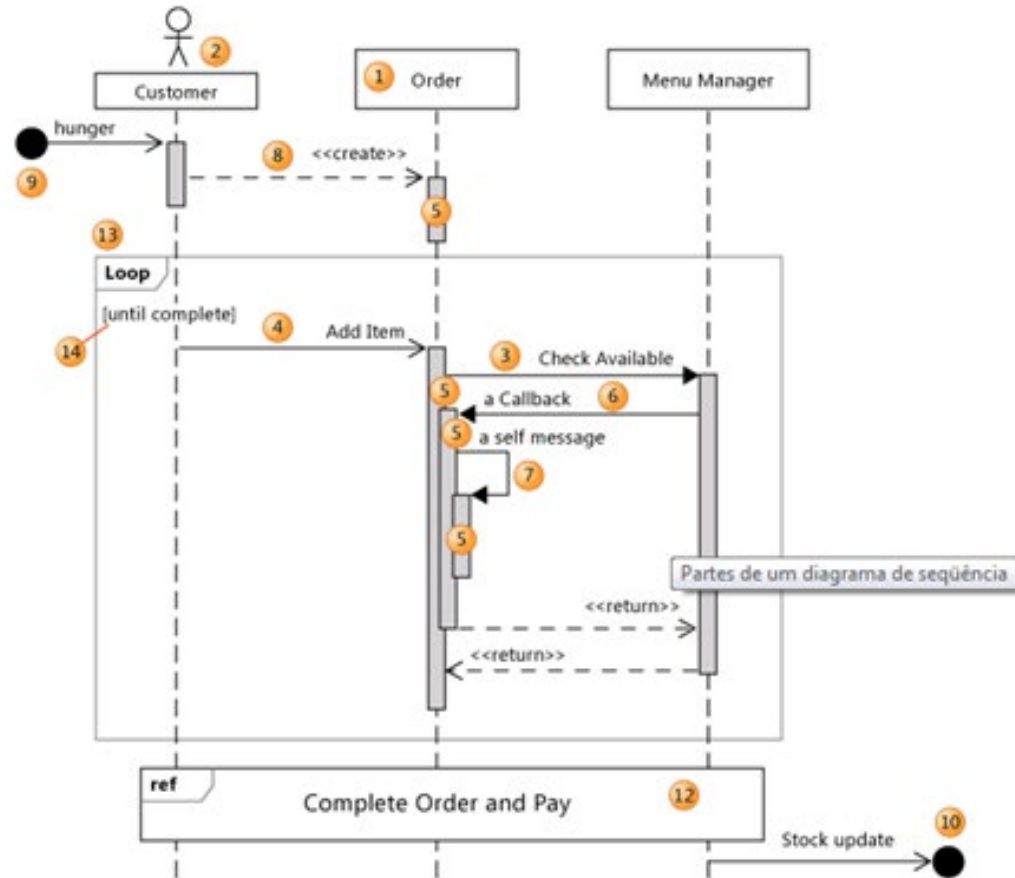


## Questão 4

- a) classe, caso de uso; um foco de controle; classe.
- b) um objeto, uma linha de vida; um foco de controle; uma mensagem.
- c) uma linha de vida; um objeto; uma mensagem, um caso de uso.
- d) um objeto; uma linha de vida; caso de uso; um foco de controle.
- e) um objeto; uma linha de vida; uma mensagem; uma mensagem.

# Questão 5

5. Analise a figura e assinale a alternativa correspondente:



Fonte: MSDN, Microsoft. Artigo Diagramas de sequência UML: referência. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409377.aspx>>. Acesso em: 20 maio 2016.



## Questão 5

Os itens 9 e 10 são, respectivamente:

- a) Mensagem encontrada / mensagem perdida
- b) Retorno de chamada/ mensagem síncrona.
- c) Mensagem assíncrona/ mensagem síncrona.
- d) Mensagem self/ mensagem de retorno.
- e) Mensagem self/ mensagem self.



# Gabarito

## 1. Resposta: B.

Comentário: todos os elementos referem-se à base do diagrama de classes, a própria classe.

I – Representa um tipo de pessoa, um lugar ou algo sobre o qual o sistema deve capturar e armazenar informações.

II – Tem o nome em negrito e centrado no seu comportamento superior.

III – Tem uma lista de atributos no seu compartimento inferior.

IV – Não mostra explicitamente as operações que estão disponíveis para todas as classes.

## 2. Resposta: A.

Comentário: todas as afirmações referem-se a métodos.

I – Representa as ações ou funções que uma classe pode realizar.

II – Pode ser classificado como uma operação construtora, de consulta ou de atualização.

III – Inclui parênteses que podem conter parâmetros especiais ou informações necessárias para que a operação seja realizada.

### 3. Resposta: C.

Comentário: Considere a explicação dos elementos do diagrama abaixo:

Observe que o item de número 1 representa a **classe**, assim compartilhando objetos comportamentais e estruturais e também a sua representação, por isso também pode ser chamada de **classificador**. Atores e casos de uso também são classificadores. O elemento 2 nada mais é do que um item que permite recolher ou **expandir** o controle para visualizar detalhes da classificação (+).

O item de número 3 é um **atributo** da classe Order, com visibilidade pública, ou seja, pode ser acessado por outros objetos. Já o item 4

desse diagrama representa uma operação, ou **método**. No caso, na representação do método 4, temos um que é AddItem (MenuItem) e o outro que é DeletarItem (MenuItem), ambos com **visibilidade** pública (+) e pertencentes à classe MenuItem, compartilhados com a classe Order por meio de agregação por composição (representada pelo losango preenchido). O item 5 representa uma associação entre as classes indicadas. Daí surgem duas novas associações, a 5a, que é por **agregação**, e a 5b, que é por **composição**.

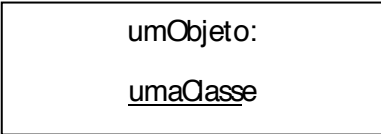

O elemento 6 representa o nome da associação. Já o elemento 7 representa, respectivamente o nome da função, muito utilizado para referenciar um objeto, apontar a ele. Uma função pode ter suas próprias características e propriedades.



# Gabarito

Elemento de número 8 indica a questão da multiplicidade, já estudada no tema anterior. O item 9 representa uma **generalização** ou herança.



## 4. Resposta: B.

Termo e Definição	Símbolo
<p>Um Objeto</p> <ul style="list-style-type: none"><li>• Participa em uma sequência enviando e/ou recebendo mensagens.</li><li>• É colocado ao longo da parte superior do diagrama.</li></ul>	
<p>Uma linha de vida</p> <ul style="list-style-type: none"><li>• Indica a vida de um objeto durante uma sequência.</li><li>• Contém um X no ponto em que a classe deixa de interagir.</li></ul>	





# Gabarito

<p>Um foco de controle</p> <ul style="list-style-type: none"><li>• É um retângulo longo e estreito colocado sobre uma linha de vida.</li><li>• Indica quando um objeto está enviando ou recebendo mensagens.</li></ul>	
<p>Uma mensagem</p> <ul style="list-style-type: none"><li>• Transmite informações de um objeto para o outro.</li></ul>	<p>umaMensagem()</p> 

## 5. Resposta: A.

Comentário: são respectivamente:

1. Mensagem encontrada: uma mensagem desconhecida ou de um comportamento desconhecido ou externo ao programa.

a) Mensagem perdida: ocorre quando não há a especificação de um objeto de envio.



# Unidade 8

## Diagrama de Estado Comportamental

---

### Objetivos

Com esta disciplina, temos os objetivos de:

1. Apresentar o que são os diagramas de estado comportamental e seus respectivos elementos componentes.
2. Proporcionar que o aluno conheça e saiba elaborar um diagrama de estado comportamental em projetos de análise e desenvolvimento de sistema sob a ótica de uma metodologia ágil.
3. Pretende-se que o aluno saiba analisar e avaliar se o diagrama de estado elaborado atende às especificações realizadas e está em consonância com os demais diagramas de caso de uso, classe e sequência.



## Introdução

Primeiramente, vamos apresentar quais são os dois tipos de diagramas de estados que podemos trabalhar. O diagrama de estados é considerado dinâmico, pois pode representar diversas mudanças de estado de um determinado objeto:

- a) Diagrama de máquina de estados comportamental: em que há a apresentação das mudanças de comportamento do sistema, de acordo com a alteração do status do objeto;
- b) diagrama de estado de protocolo: utiliza o ciclo de vida de um objeto ou de um classificador do sistema.

Link



Acesse o link: <[http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/uml/diagramas/estado/diag\\_estados.htm](http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/uml/diagramas/estado/diag_estados.htm)> e aprenda mais sobre os diagramas de estados e como desenvolvê-los mais facilmente. Acesso em: 22 maio 2016.

Os diagramas de estado evidenciam a mudança que pode ocorrer nas classes do sistema a considerar o seu ciclo de vida:



O diagrama de estados comportamental mostra os diferentes estados pelos quais uma determinada instância da classe passa durante sua existência em resposta a eventos, com as respostas e ações. (DENNIS et al., 2014, p. 524).

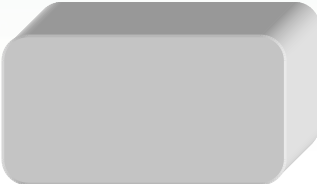

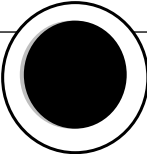
É mais utilizado para que se possa obter uma visão mais completa, de como ocorre a mudança comportamental ou de **status** de uma determinada classe. Dessa forma, busca fornecer informações que facilitem o desenvolvimento de regras para os algoritmos dos métodos de uma determinada classe:



Um estado é um conjunto de valores que descreve um objeto em um instante específico no tempo e representa um momento na vida de um objeto em que ele satisfaz alguma condição, executa alguma ação ou aguarda algo ocorrer. (DENNIS et al., 2014, p. 520).

Observe no quadro abaixo a apresentação de cada um dos elementos de um diagrama de estados:

Quadro 14 – Elementos de um diagrama de estados

Termo e Definição	Símbolo
<b>Um estado</b> <ul style="list-style-type: none"><li>• É mostrado como um retângulo com cantos arredondados.</li><li>• Tem um nome que representa o estado de um objeto.</li></ul>	
<b>Um estado inicial</b> <ul style="list-style-type: none"><li>• É mostrado com um pequeno círculo sólido preenchido.</li><li>• Representa o ponto no qual um objeto começa a existir.</li></ul>	
<b>Um estado final</b> <ul style="list-style-type: none"><li>• É mostrado como um círculo circunscrevendo um pequeno círculo sólido preenchido (bull's eye, ou alvo).</li></ul>	

### Um evento

- É uma ocorrência notável que aciona uma mudança de estado.
- Pode ser uma condição especificada que se torne verdadeira, o recebimento de um sinal explícito de um objeto por outro, ou a passagem de um período especificado.
- É usado para identificar uma transição.

### Nome do evento

### Uma transição

- Indica que um objeto no primeiro estado ingressará no segundo estado.
- É acionada pela ocorrência do evento identificado na transição.
- É mostrada como uma seta sólida de um estado para outro, identificada pelo nome do evento.



Fonte: DENNIS et. al, 2014, p. 521.



## Para saber mais


Leia o material didático disponível em: <<http://www.dca.fee.unicamp.br/~gudwin/ftp/ea976/AtEst.pdf>> e obtenha mais informações sobre os diagramas de atividades e o de estado. Acesso em: 22 maio 2016.

Um evento está associado a um objeto e, quando ocorre, há a alteração do valor do objeto, então, como consequência dessa mudança, há também uma mudança no estado do objeto. Vários podem ser os motivos que causam a mudança de um valor do objeto, por exemplo, uma condição pode ter sido atendida, ou, ainda, a execução de um determinado método que gerou a alteração do valor do objeto, e assim por

diante. Dessa forma, o estado do objeto tem por função determinar uma resposta que permita identificar o seu novo *status*.

## Para saber mais

Leia “Utilizando UML: Diagrama de máquina de estados”. Disponível em: <<http://www.devmedia.com.br/artigo-sql-magazine-65-utilizando-uml-diagrama-de-maquina-de-estados/13372>>. Acesso em: 22 maio 2016.



As setas servem para indicar o fluxo da alteração de estados e também representam as transições de um estado para o outro:



Transição é um relacionamento que representa o movimento do objeto de um estado para outro. Algumas transições terão uma condição de ocorrência (guard condition, também denominada condição de segurança, restrição de iteração ou ainda condição de guarda) (DENNIS et al., 2014, p. 521).

A condição mencionada por Dennis (et al., 2014), considera apenas expressões chamadas **booleanas**, pois indicam se uma determinada ação ou método do sistema atendeu às únicas duas situações permitidas em testes de verificação booleanas: se é verdadeiro ou falso.



Figura 21 – Pseudo-escolha: condição



Fonte: Revista BW, 2015. Disponível em: <<http://www.revistabw.com.br/revistabw/uml-diagrama-de-estados/>>. Acesso em: 23 maio 2016.

Nesse caso, somente há a troca de estado, se atendeu, ou seja, é verdadeira a informação resultante, no que tange atender ou não a uma determinada regra ou condição.



Artigo muito importante sobre diagrama de estados, publicado na *Revista Brasileira da Web*: <<http://www.revistabw.com.br/revistabw/uml-diagrama-de-estados/>>. Acesso em: 23 maio 2016.

## 1. Criando Diagramas de Estado

Considere que você precisa desenvolver o diagrama de estado de uma classe que foi utilizada no diagrama de classes para conseguir compreender melhor como ela funciona, entender o que provoca as mudanças de estado, que tipo de eventos são esses. Então, o primeiro passo é: **identificar os Estados** que essa classe terá, sempre observando o seu ciclo de vida.

O ciclo de vida de uma classe torna-se uma informação importante, pois é preciso saber e documentar as ocorrências desse período, e, principalmente, o que levou às alterações de Estado. Se tiver criado o diagrama de caso de uso dessa classe, pode se respaldar por ele para conseguir realizar o próximo passo que é: **identificar as transições**.

### Para saber mais

Estude sobre as convenções de um diagrama de estados em: <[https://technet.microsoft.com/pt-br/library/cc756951\(v=ws.10\).aspx](https://technet.microsoft.com/pt-br/library/cc756951(v=ws.10).aspx)>.

Acesso em: 23 maio 2016.

Figura 22 – Exemplo de transição entre estados



Fonte: Revista BW, 2015. Disponível em: <<http://www.revistabw.com.br/revistabw/uml-diagrama-de-estados/>>. Acesso em: 23 maio 2016.

Existe ainda a possibilidade de, após a execução, um método ou ação do sistema retornar ao estado de origem. A essa transição dá-se o nome de autotransição:

Figura 23 – Autotransição de estados



Fonte: Revista BW, 2015. Disponível em: <<http://www.revistabw.com.br/revistabw/uml-diagrama-de-estados/>>. Acesso em: 23 maio 2016.

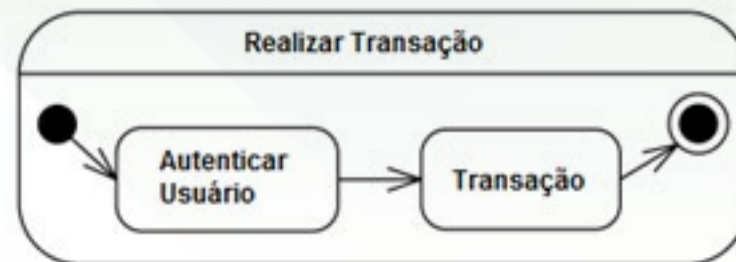
O diagrama de caso de uso ajuda nesse desenvolvimento, pois, para ser elaborado, é preciso que o analista tenha em mãos ou já tenha bem definidas as etapas que aquele cenário deverá executar.

As atividades realizadas por um estado, de acordo com a *Revista BW* (2015), compreendem o que chamamos de:

- *on entry*: que representa o comportamento ou a execução de uma função, quando se entra em um determinado estado;
- *on exit*: que determina qual foi a última função executada antes do estado ser alterado;
- *do action*: que representa a atividade que está sendo executada durante o período em que o objeto permanece em um estado.

Além dessas asserções, a *Revista BW* (2015) também apresenta um diagrama de estado composto. Observe como é:

Figura 24 – Estado composto



Fonte: Revista BW, 2015. Disponível em: <<http://www.revistabw.com.br/revistabw/uml-diagrama-de-estados/>>. Acesso em: 23 maio 2016.

O diagrama apresentado na Figura 24 representa um estado que contém outros estados, ou seja, é composto por outros estados em sua estrutura. Esses estados que estão contido no Estado maior são chamados de subestados.



Assista ao vídeo que ensino como fazer o diagrama de estados e explica suas propriedades. Disponível em: <[https://www.youtube.com/watch?v=mJHT\\_b26rCs](https://www.youtube.com/watch?v=mJHT_b26rCs)>.

Acesso em: 23 maio 2016.



## Glossário

**Status:** refere-se à forma como um objeto está e se sofre alterações em suas características, ou valores. Também indica posição.

**Pseudo:** remete a algo artificial, que ocorreu, mas não foi mantido, como o que acontece no caso de uma pseudotransição, em que a ação do sistema retorna a origem.

**Booleanas:** refere-se a um tipo de dado primitivo que permite a verificação de apenas dois estados: verdadeiro ou falso. Pode-se aplicar a diversas situações computacionais.



# Questão para reflexão

Após os estudos sobre os quatro diagramas fundamentais da UML, de que forma podem ser implementados nas empresas em conjunto com a visão de gestão de projetos e a adoção de uma metodologia ágil?







## Considerações Finais

Primeiramente, foram identificados os dois tipos de diagramas de estados que podemos desenvolver. Em seguida, foram apresentados os seus componentes e respectivas descrições. Após essas abordagens, o material buscou evidenciar o que são eventos que podem ocorrer e levar à mudança de um estado em instâncias de uma classe, ou seja, em um determinado objeto. Para finalizar, foram apresentadas algumas formas de desenvolver e analisar um diagrama de estados.



## Referências

DENNIS, Alan et al. **Análise e Projeto de Sistemas**. 5. ed. Rio de Janeiro: LTC, 2014.

REVISTABW. UML: Diagrama de Estados. **Revista Brasileira de Web**. Disponível em: <<http://www.revistabw.com.br/revistabw/uml-diagrama-de-estados/>>. Acesso em: 23 maio 2016.



# Assista a suas aulas



## Aula 8 - Tema: Diagrama de Estado Comportamental - Bloco I

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/8e1979422609ef300beb4d5f360704ca>>.



## Aula8-Tema:DiagramadeEstadoComportamental - Bloco II

Disponível em: <<http://fast.player.liquidplatform.com/pApiv2/embed/dbd3957c747affd3be431606233e0f1d/b6b28e6fa01ec1a116744e215e5557bb>>.



# Questão 1

**1. Elenque, ao lado das descrições, o seu respectivo conceito associado, disponível em apenas uma das alternativas abaixo:**

Definição	Termo
• É mostrado como um retângulo com cantos arredondados.	
• É mostrado com um pequeno círculo sólido preenchido.	
• É mostrado como um círculo circunscrevendo um pequeno círculo sólido preenchido ( <i>bull's eye</i> , ou alvo).	
• Pode ser uma condição especificada que se torne verdadeira, o recebimento de um sinal explícito de um objeto por outro, ou a passagem de um período especificado.	
• Indica que um objeto no primeiro estado ingressará no segundo estado.	

- a) Uma transição; um estado; um evento; uma transição; uma pseudotransição.
- b) Uma transição; um estado; um estado inicial; uma transposição; uma pseudotransição.
- c) Um estado; um estado final; um evento; uma transição; uma pseudotransição.
- d) Um estado; um estado inicial; um estado final; um evento; uma transição.
- e) Uma transição; um estado; um evento; uma transição; um evento.



## Questão 2

### 2. Está correto o que se afirma em:

I – Um evento está associado a um objeto e, quando ocorre, há a alteração do valor do objeto.

II – Os estados são modificados apenas se o evento for uma condição atendida e um retorno ao usuário for estabelecido.

III – Quando ocorre um evento, há a alteração do valor do objeto, então, como consequência dessa mudança, há também uma mudança no estado do objeto.

Está correto apenas o que se afirma em:

- a) I e II;
- b) I e III;
- c) I, II e III;
- d) Apenas na afirmação I;
- e) Apenas na afirmação III.



## Questão 3

**3. Complete as lacunas com as palavras dispostas em apenas uma das alternativas abaixo:**

I – \_\_\_\_\_: que representa o comportamento ou a execução de uma função, quando se entra em um determinado estado.

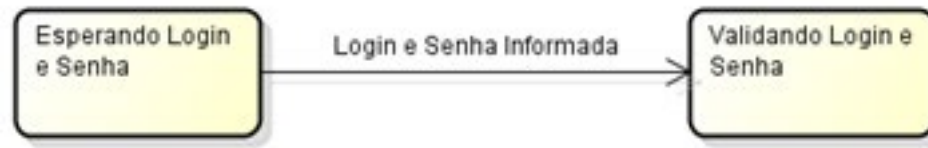
II – \_\_\_\_\_: que determina qual foi a última função executada antes de o estado ser alterado.

III – \_\_\_\_\_: que representa a atividade que está sendo executada durante o período em que o objeto permanece em um estado.

- a) *On entry / on exit / do action*
- b) *On entry / do action / do exit*
- c) *Do action / on entry / on exit*
- d) *On entry / do action / do action*
- e) *On entry / on exit / do action*

## Questão 4

4. O diagrama de estados abaixo representa qual etapa?

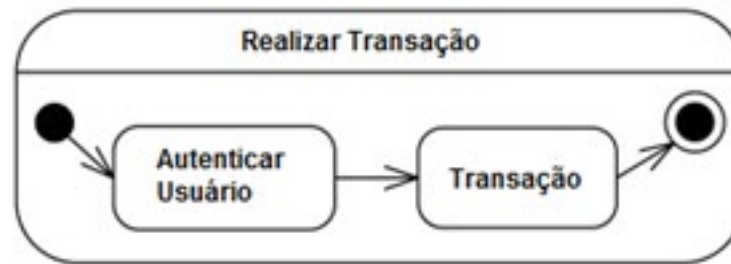


Fonte: Revista BW, 2015. Disponível em: <<http://www.revistabw.com.br/revistabw/uml-diagrama-de-estados/>>. Acesso em: 23/05/2016.

- a) Diagrama de classe e suas associações.
- b) Diagrama de estado representando a transição entre estados.
- c) Diagrama de caso de uso e seus atores.
- d) Diagrama de estado e uma pseudotransição.
- e) Diagrama de estados em uma verificação de condição.

## Questão 5

5. Qual é o conceito empregado na figura abaixo? Analise e assinale a alternativa correspondente:



Fonte: Revista BW, 2015. Disponível em: <<http://www.revistabw.com.br/revistabw/uml-diagrama-de-estados/>>. Acesso em: 23 maio 2016.

- a) Diagrama de transição.
- b) Diagrama de estado de transição
- c) Diagrama de estado em verificação de condição.
- d) Diagrama de estado composto.
- e) Diagrama de estado de autenticação.





# Gabarito

## 1. Resposta: D.

Definição	Termo
• É mostrado como um retângulo com cantos arredondados.	Um Estado
• É mostrado com um pequeno círculo sólido preenchido.	Um Estado inicial
• É mostrado como um círculo circunscrevendo um pequeno círculo sólido preenchido ( <i>bull's eye</i> , ou alvo).	Um Estado final
• Pode ser uma condição especificada que se torne verdadeira, o recebimento de um sinal explícito de um objeto por outro, ou a passagem de um período especificado.	<b>Um evento</b>
• Indica que um objeto no primeiro estado ingressará no segundo estado.	Uma transição

## 2. Resposta: B.

Comentário: um evento está associado a um objeto e, quando ocorre, há a alteração do valor do objeto, então, como consequência dessa mudança, há também uma mudança no estado do objeto. Vários podem ser os motivos que causam a mudança de um valor do objeto, por exemplo, uma condição pode ter sido atendida, ou, ainda, a execução de um determinado



# Gabarito

método que gerou a alteração do valor do objeto, e assim por diante. Dessa forma, o estado do objeto tem por função determinar uma resposta que permita identificar o seu novo *status*.

## 3. Resposta: E.

I – *Do entry*: que representa o comportamento ou a execução de uma função, quando se entra em um determinado estado.

II – *Do exit*: que determina qual foi a última função executada antes de o estado ser alterado.

III – *Do action*: que representa a atividade que está sendo executada durante o período em que o objeto permanece em um estado.

## 4. Resposta: B.

Comentário: O ciclo de vida de uma classe torna-se uma informação importante, pois é preciso saber e documentar as ocorrências desse período, e, principalmente, o que levou às alterações de Estado. Se tiver criado o diagrama de caso de uso dessa classe, pode se respaldar por ele para conseguir realizar o próximo passo que é: **identificar as transições**.

Figura 25 – Exemplo de transição entre estados



Fonte: Revista BW, 2015. Disponível em: <<http://www.revistabw.com.br/revistabw/uml-diagrama-de-estados/>>. Acesso em: 23 maio 2016.

## 5. Resposta: D.

Comentário: o diagrama apresentado na figura representa um estado que contém outros estados, ou seja, é composto por outros estados em sua estrutura. Esses estados que estão contido no Estado maior são chamados de subestados.

