

DEVOPS & AGILE CULTURE

# DESVENDANDO O UML

CRISTINA B. MATOS E ELISA M. SUEMASU



4

## LISTA DE FIGURAS

Figura 4.1 – Exemplo de diagrama de fluxo de dados nível 0.....	7
Figura 4.2 – Histórico da orientação de objetos até o surgimento da UML .....	8
Figura 4.3 – História UML.....	9
Figura 4.4 – Representação visual .....	9
Figura 4.5 – Diagrama de caso de uso.....	10
Figura 4.6 – Diagramas UML, versão 2.5.1 .....	11
Figura 4.7 – Diagramas Estáticos e Dinâmicos .....	12
Figura 4.8 – Exemplo de Diagrama de Atividades .....	14
Figura 4.9 – Exemplo de Diagrama de Atividades com raias ou partições.....	15
Figura 4.10 – Exemplo de Diagrama de Atividades, visão caso de uso .....	16
Figura 4.11 – Exemplo de Diagrama de Caso de Uso referente ao sistema iLikeBeer .....	17
Figura 4.12 – Exemplo de Diagrama de Caso de Uso referente ao processo Reservar Mesa .....	18
Figura 4.13 – Exemplo de Diagrama de Classe referente ao processo Reservar Mesa com notação de Pacotes .....	19
Figura 4.14 – Exemplo de Diagrama de Classe referente ao processo de Pedido ...	19
Figura 4.15 – Exemplo de Diagrama de Sequência de um processo de Pedido.....	20
Figura 4.16 – Exemplo de Diagrama de Estado .....	21
Figura 4.17 – Exemplo de Diagrama Máquina de Estados .....	21
Figura 4.18 – Exemplo de Diagrama de Componentes, notação UML 1.4.....	22
Figura 4.19 – Exemplo de Diagrama de Componente .....	23
Figura 4.20 – Exemplo de Diagrama de Implantação com os componentes relacionados .....	23
Figura 4.21 – Exemplo de Diagrama de Objetos.....	24

## LISTA DE QUADROS

Quadro 4.1 – Modelagem de sistema orientado a objetos na década de 1990.....7

EUROPE

## SUMÁRIO

4 UML – PARTE 1 .....	5
4.1 Conceito de UML.....	5
4.2 Histórico de desenvolvimento de sistemas.....	5
4.3 Histórico UML.....	7
4.4 Mas, afinal, o que é e o que não é UML? .....	10
4.5 Quais são as razões para construção de modelos visuais? .....	11
4.6 Diagramas da UML.....	11
4.6.1 Diagramas Dinâmicos: .....	12
4.6.2 Diagramas Estáticos:.....	13
4.6.3 Diagrama de Atividades .....	13
4.6.4 Diagrama de Caso de Uso .....	16
4.6.5 Diagrama de Classe .....	18
4.6.6 Diagrama de Sequência .....	20
4.6.7 Diagrama de Máquina de Estados .....	20
4.6.8 Diagrama de Componentes.....	22
4.6.9 Diagrama de Implantação .....	23
4.6.10 Diagrama de Objetos.....	23
4.6.11 Diagrama de Colaboração.....	24
4.6.12 Diagrama de Pacotes .....	25
4.6.13 Diagrama de Perfil.....	25
REFERÊNCIAS .....	27

## 4 DESVENDANDO O UML

O desenvolvimento de software é uma atividade bastante complexa, e em razão disso, existem dificuldades e surgem vários problemas quando não há um padrão comum de interpretação para a equipe envolvida. Para tanto, é necessário desenvolver o software sem que haja um retrabalho, cuja manutenção no futuro seja fácil e no qual todos os envolvidos interpretem a mesma linguagem. A UML propõe a construção de diagramas para o desenvolvimento de sistemas orientados a objetos a fim de facilitar, por meio de notação visual, uma representação comum em determinado momento do desenvolvimento de software.

Este capítulo apresenta um breve histórico a respeito da visão de análise de sistema até chegar ao surgimento da UML como melhor proposta para a representação na orientação objeto.

### 4.1 Conceito de UML

Conforme Guedes (2011, p.8) afirma:

*A UML – Unified Modeling Language ou Linguagem de Modelagem Unificada – é uma linguagem visual utilizada para modelar softwares baseados no paradigma da orientação objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios da aplicação. Esta linguagem tornou-se, nos últimos anos, a linguagem padrão de modelagem adotada internacionalmente pela indústria de engenharia de software.*

A UML é uma linguagem com padrões visuais para modelar sistemas orientados a objetos.

### 4.2 Histórico de desenvolvimento de sistemas

Nas décadas de 1950 e de 1960, o desenvolvimento de sistema era tecnicista, ou seja, o usuário solicitava ao programador um sistema, e este, somente com a visão da necessidade estabelecida, programava sem metodologia ou padrão, tudo era feito *ad hoc*. De acordo com Bezerra (2007, p. 13), “talvez o uso deste termo denote a

abordagem da primeira fase de desenvolvimento de sistema, no qual não havia planejamento inicial”.

A análise foi concebida na década de 1970 e surgiu com o objetivo de construir sistemas mais eficientes e eficazes, por meio do uso de modelos e padrões estruturados para facilitar o entendimento da necessidade do usuário, pois antes do uso da análise, os sistemas não atendiam adequadamente a real necessidade dos usuários, além dos projetos que extrapolavam os prazos e os orçamentos por falta de entendimento da necessidade do usuário, o “ouvir” e programar sem estudo prévio causava grandes problemas no desenvolvimento de sistemas. Portanto, antes da análise não havia padrões, ferramentas, técnicas nem métodos para o desenvolvimento de sistemas.

De acordo com Bezerra (2007, p. 13):

*O rápido crescimento da capacidade computacional das máquinas resultou na demanda por sistemas de software cada vez mais complexos. Portanto, o surgimento destes sistemas complexos resultou na necessidade de reavaliação das formas de desenvolvimento de sistema, consequentemente, o surgimento de metodologias, práticas, técnicas e padrões para a modelagem de sistema.*

Na década de 1980, com o surgimento dos computadores pessoais, os PCs, aumentou a necessidade de desenvolvimento de sistemas mais complexos, a análise estruturada se consolidou na primeira metade dessa década.

Em 1980, Edward Yourdon lançou o livro *Análise Estruturada Moderna*, que se tornou referência neste assunto.

A análise estrutural foi utilizada para modelagem de sistemas estruturados por meio da representação gráfica de Diagramas de Fluxos de Dados, por exemplo.

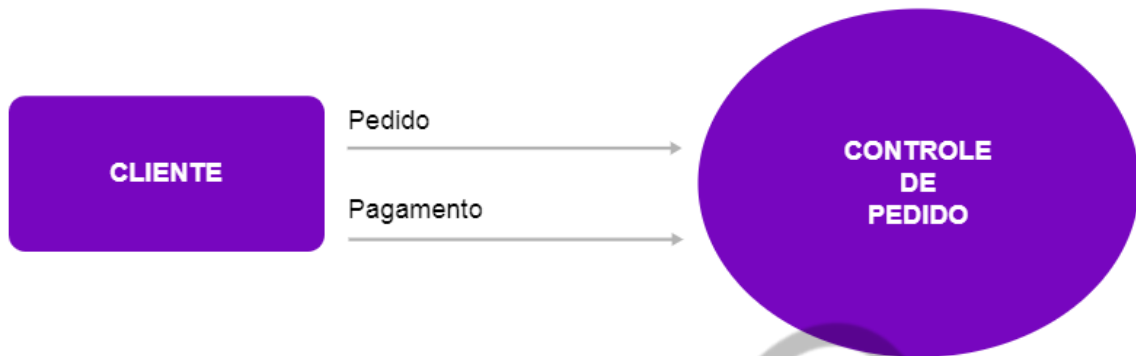


Figura 4.1 – Exemplo de diagrama de fluxo de dados nível 0  
 Fonte: Elaborado pela autora Cristina (2016)

Na década de 1990, surgiu um novo paradigma para a modelagem de sistemas orientados a objetos. Para essa década, as principais técnicas para modelagem de sistemas são: OOSE (*Object Oriented Software Engineering*), de Ivair Jacobson; o OMT (*Object Modeling Technique*), de James Rumbaugh; e o método de Grady Booch.

Técnica de modelagem	Descrição
Ivair Jacobson - <b>OOSE</b>	Ênfase em casos de uso - descrição do cenário. Demonstrar interação do usuário com o sistema.
James Rumbaugh - <b>OMT</b>	Representação de objetos, classes e relacionamentos.
Grady Booch - <b>Booch</b>	Sistema deve ser analisado por meio de várias visões, para cada uma deve haver um diagrama.

Modelagem de sistema orientado a objetos na década de 1990

Quadro 4.1 – Modelagem de sistema orientado a objetos na década de 1990  
 Fonte: Elaborado pela autora Cristina (2016)

### 4.3 Histórico UML

Entre 1991 e 1995, começou o desenvolvimento de técnicas para modelagem de sistemas orientados a objetos, sendo os principais contribuintes: Ivair Jacobson, James Rumbaugh e Grady Booch, os quais reaproveitaram as melhores

características e artefatos das técnicas propostas (conforme Figura Histórico da orientação de objetos até o surgimento da UML) para o desenvolvimento da UML. Inicialmente era chamada de UM (*Unified Method*), somente com a entrada de Booch, quando os três trabalharam na empresa Rational Software, as três propostas foram unificadas e surgiu realmente a UML (*Unified Modeling Language*).

Em 1997, a OMG (*Object Management Group*), uma organização que define e ratifica a padronização de assuntos relacionados à orientação de objetos, aprovou e adotou o padrão UML para a modelagem de sistemas orientados a objetos, e desde então, esse padrão tem grande aceitação no mercado e na comunidade de desenvolvedores.

A UML (*Unified Modeling Language*) se tornou uma linguagem visual ou notação visual que permite representar os paradigmas da orientação a objetos para modelagem de sistemas, por meio da notação visual com os diagramas é possível representar várias perspectivas do sistema.

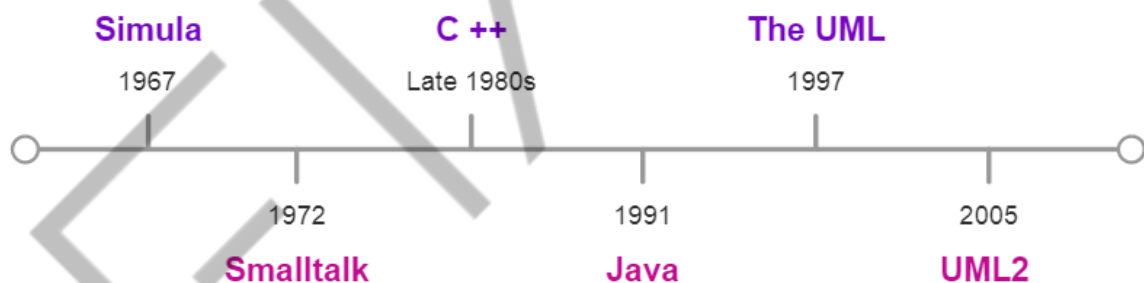


Figura 4.2 – Histórico da orientação de objetos até o surgimento da UML  
Fonte: Leandro Rubim (2010)



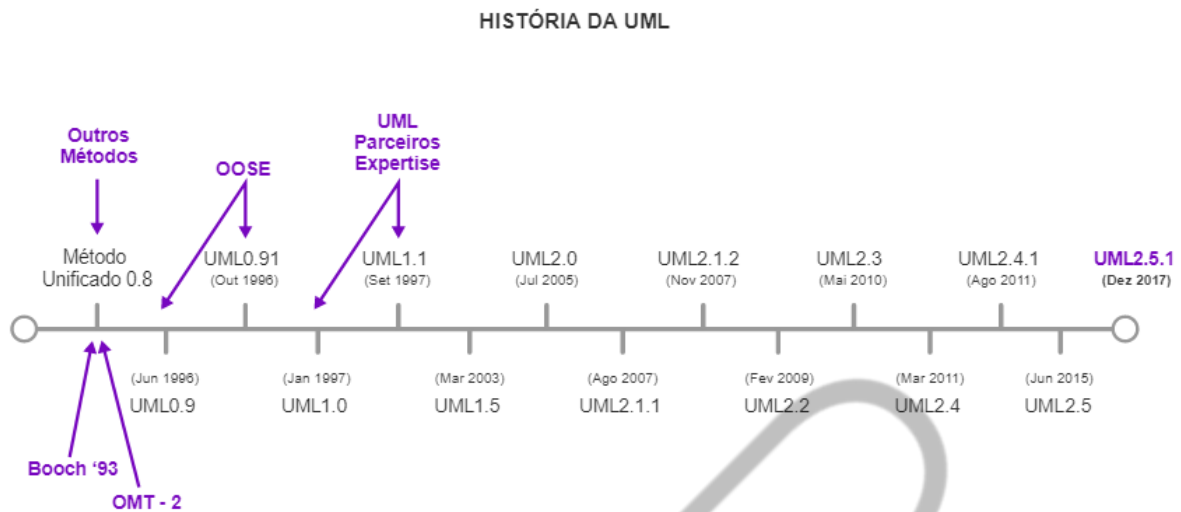


Figura 4.3 – História UML  
 Fonte: Adaptado de <http://www.omg.org/spec/UML/> (2018)

Por que utilizar diagramas para modelar sistemas?

- Os diagramas são notações visuais que facilitam a interpretação lógica do sistema em vários aspectos.
- Os diagramas fornecem uma representação objetiva do sistema. Como diz o ditado: “Uma imagem diz mais que mil palavras”.



Figura 4.4 – Representação visual  
 Fonte: Banco de imagens Shutterstock (2016)

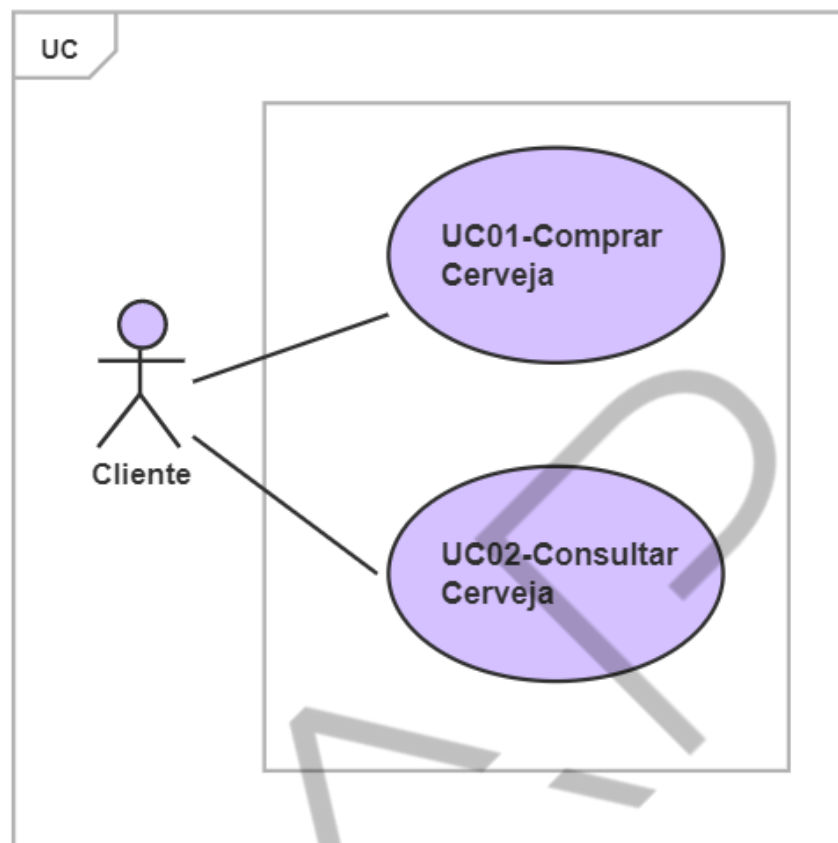


Figura 4.5 – Diagrama de caso de uso  
Fonte: Elaborado pela autora Elisa Suemasu (2016)

#### 4.4 Mas, afinal, o que é e o que não é UML?

UML é uma linguagem visual para construir, documentar e especificar sistemas orientados a objetos. Não confunda com linguagem de programação, ela não tem relação com o processo de desenvolvimento e pode ser definida como um padrão de notação visual.

Responsável por toda documentação da arquitetura do sistema, o UML especifica os requisitos e auxilia nos casos de testes, oferecendo suporte no planejamento e gerenciamento de cada versão do software.

Não confunda UML com: linguagem de programação, método ou metodologia!

#### 4.5 Quais são as razões para construção de modelos visuais?

- Facilita a comunicação entre as pessoas envolvidas no projeto de software.
- Redução do tempo e do custo de desenvolvimento.
- Futura manutenção do sistema.
- Complexidade do sistema.
- Controle da complexidade.
- Gerenciamento dos entregáveis.
- Gerenciamento de mudanças.
- Reduz as ambiguidades.

#### 4.6 Diagramas da UML

A UML utiliza o conceito de visões que pressupõe o objetivo de cada diagrama dentro do contexto de desenvolvimento de software.

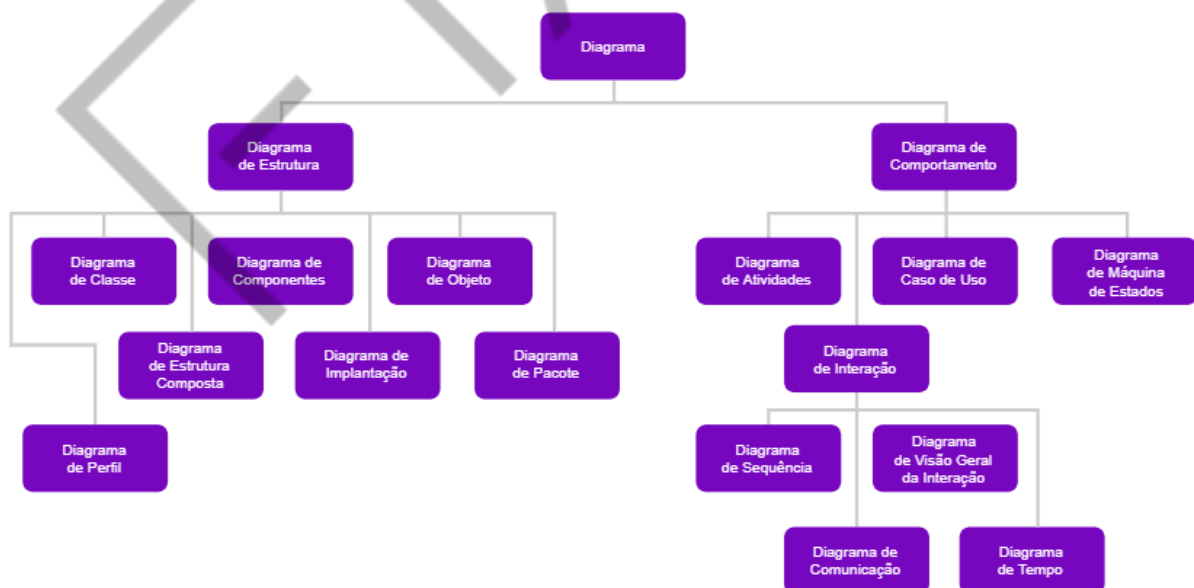


Figura 4.6 – Diagramas UML, versão 2.5.1  
Fonte: OMG (2017)

Como é mostrado na Figura Diagramas Estáticos e Dinâmicos, a seguir, na UML temos dois tipos principais de Diagramas: os diagramas estáticos e os diagramas dinâmicos.

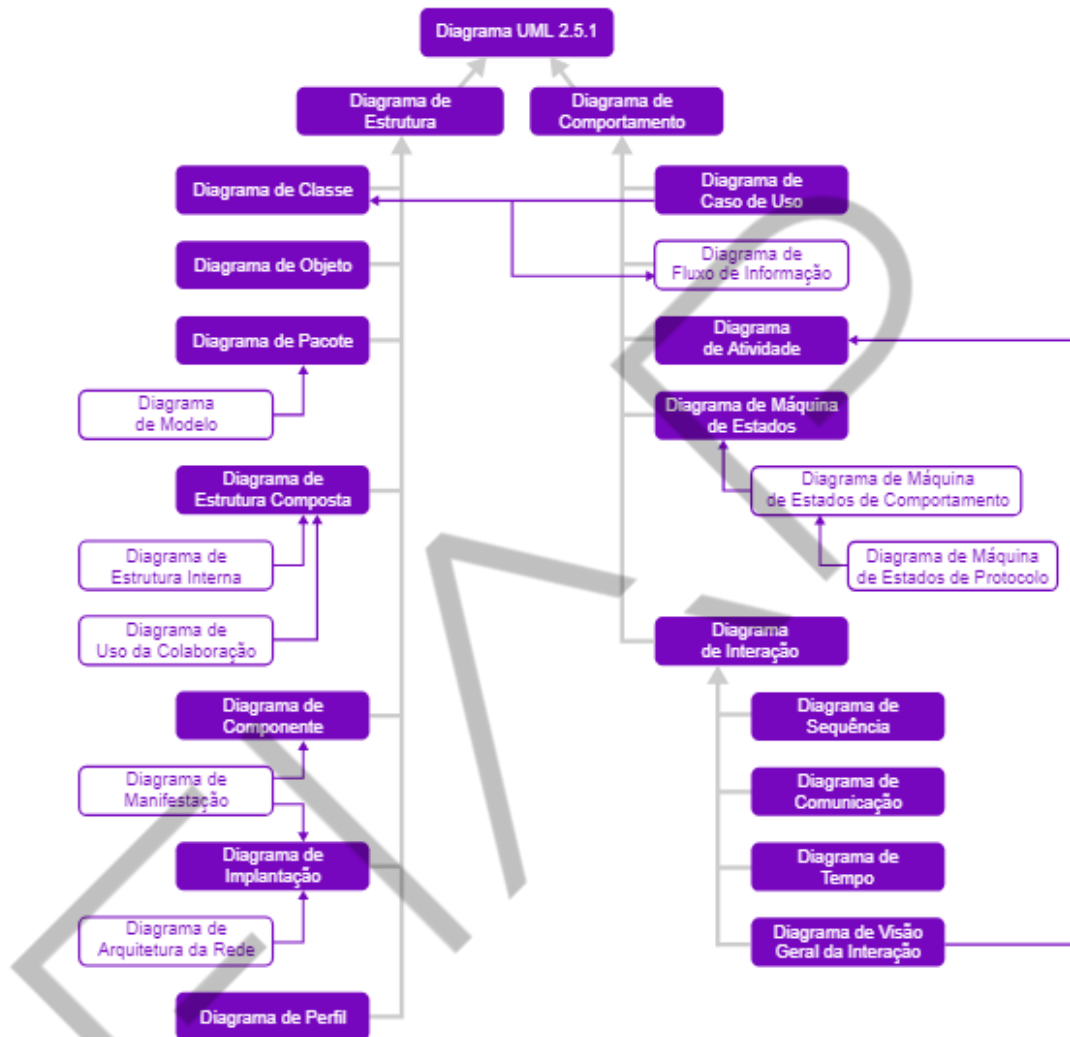


Figura 4.7 – Diagramas Estáticos e Dinâmicos  
Fonte: OMG (2017)

#### 4.6.1 Diagramas Dinâmicos

- Diagrama de Atividades
- Diagrama de Caso de Uso
- Diagrama de Fluxo de Informação
- Diagrama de Máquina de Estado
- Diagramas de Interação:

- Diagrama de Sequência
- Diagrama de Comunicação
- Diagrama de Tempo
- Diagrama de Visão Geral da Interação

#### 4.6.2 Diagramas Estáticos:

- Diagrama de Classe
- Diagrama de Objetos
- Diagrama de Pacotes
- Diagrama de Estrutura Composta
- Diagrama de Componentes
- Diagrama de Implantação
- Diagrama de Perfil

Agora vamos conhecer um pouco sobre alguns dos diagramas mais utilizados no mercado, pois nos próximos capítulos os diagramas serão explicados detalhadamente e com exemplos.

De acordo com Bezerra (2007, p. 16), “os autores da UML propõem que um sistema pode ser desenvolvido a partir de cinco visões (BOOCH *et al.*, 2006), e cada uma dá ênfase aos diferentes aspectos do sistema”.

Os diagramas que exemplificaremos a seguir trazem o objetivo e a visão propostos para cada um deles.

#### 4.6.3 Diagrama de Atividades

- **Objetivo:** é um diagrama dinâmico que representa o fluxo, a sequência de tarefas de um processamento.
- **Visão de Concorrência:** trata da divisão do sistema em processos e processadores. Permite uma melhor utilização do ambiente onde o sistema

se encontrará, se o mesmo possui exceções paralelas e se existem gerenciamentos assíncronos.

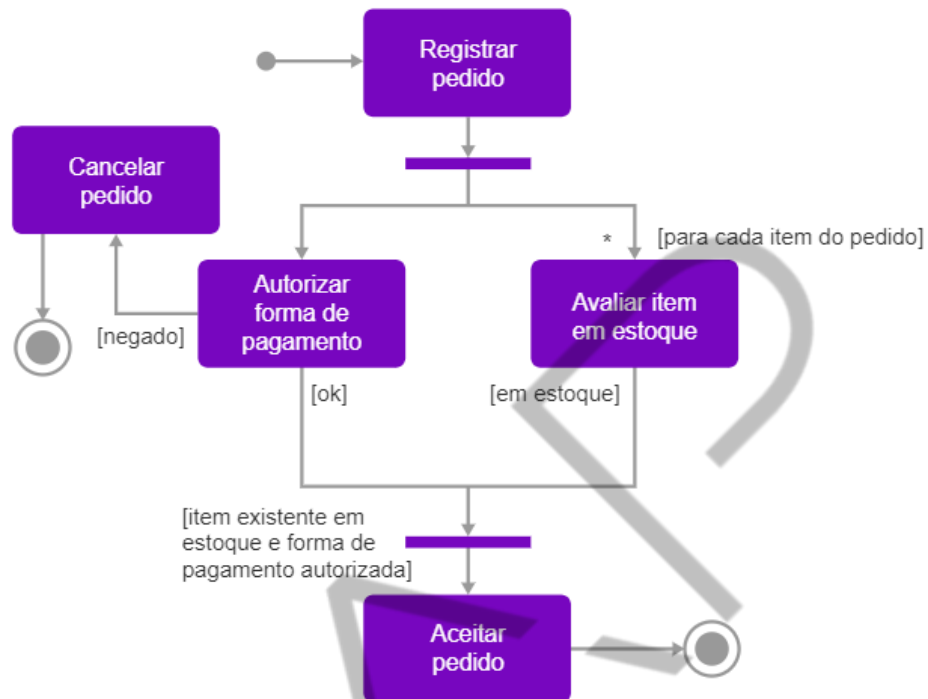


Figura 4.8 – Exemplo de Diagrama de Atividades  
Fonte: Bezerra (2007)

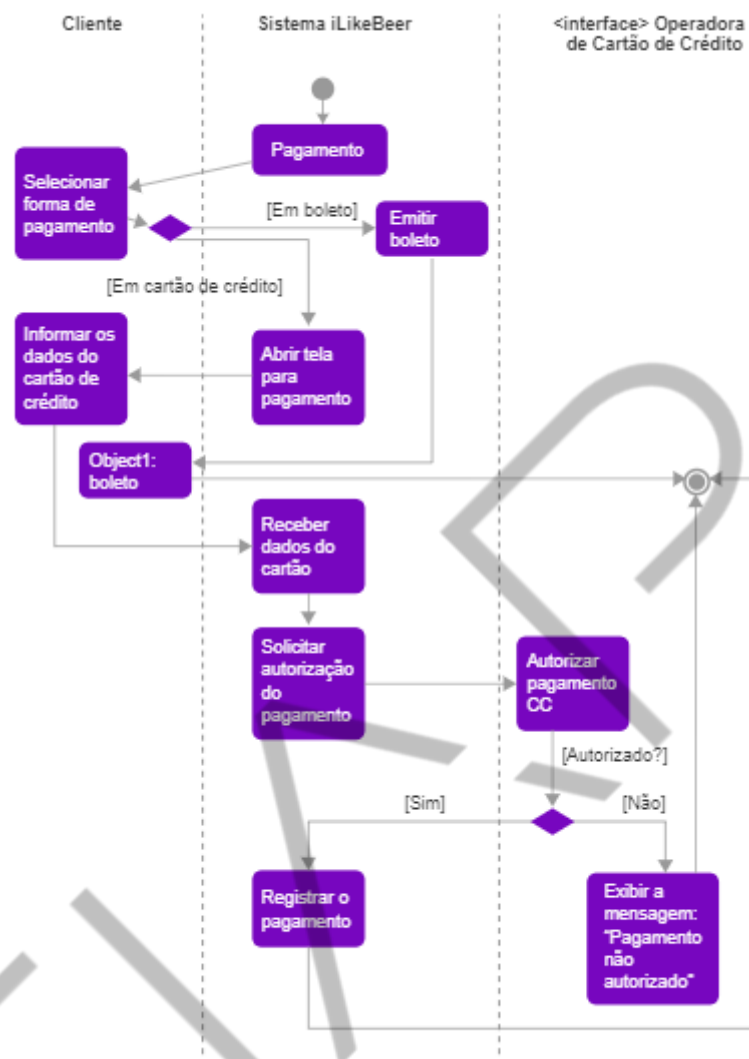


Figura 4.9 – Exemplo de Diagrama de Atividades com raias ou partições  
 Fonte: Elaborado pela autora Elisa Suemasu (2016)

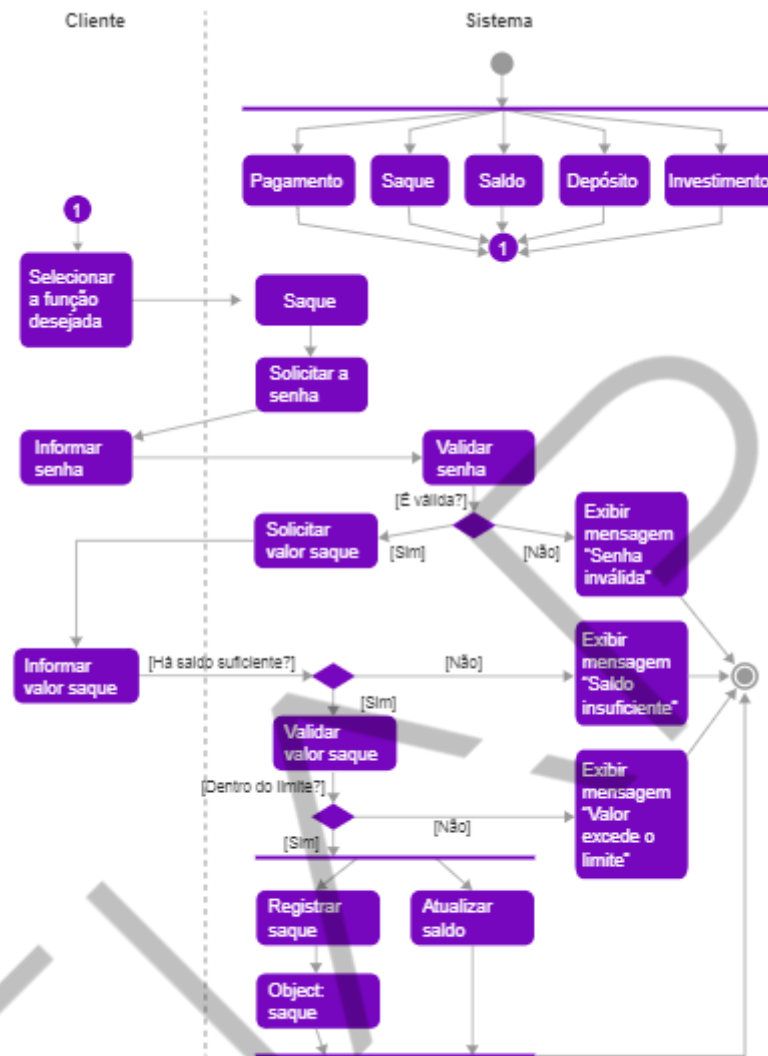


Figura 4.10 – Exemplo de Diagrama de Atividades, visão caso de uso  
 Fonte: Elaborado pela autora Elisa Suemasu (2016)

#### 4.6.4 Diagrama de Caso de Uso

- **Objetivo:** é um diagrama dinâmico que representa um conjunto de ações (casos de uso) que os sistemas devem executar em interação com um ou mais usuários externos do sistema (atores) a fim de fornecer resultados para partes interessadas do(s) sistema(s).
- **Visão “use-case”:** descreve a funcionalidade do sistema que é executada pelos atores externos (usuários).



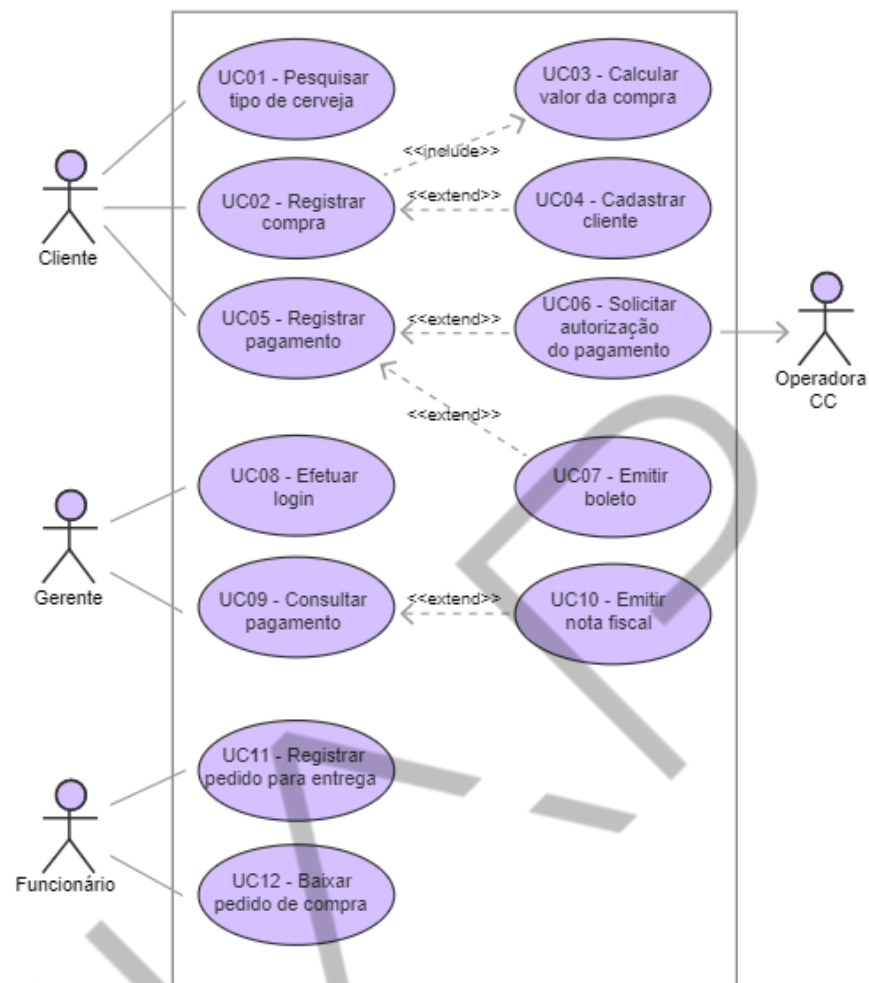


Figura 4.11 – Exemplo de Diagrama de Caso de Uso referente ao sistema iLikeBeer  
Fonte: Elaborado pela autora Elisa Suemasu (2016)

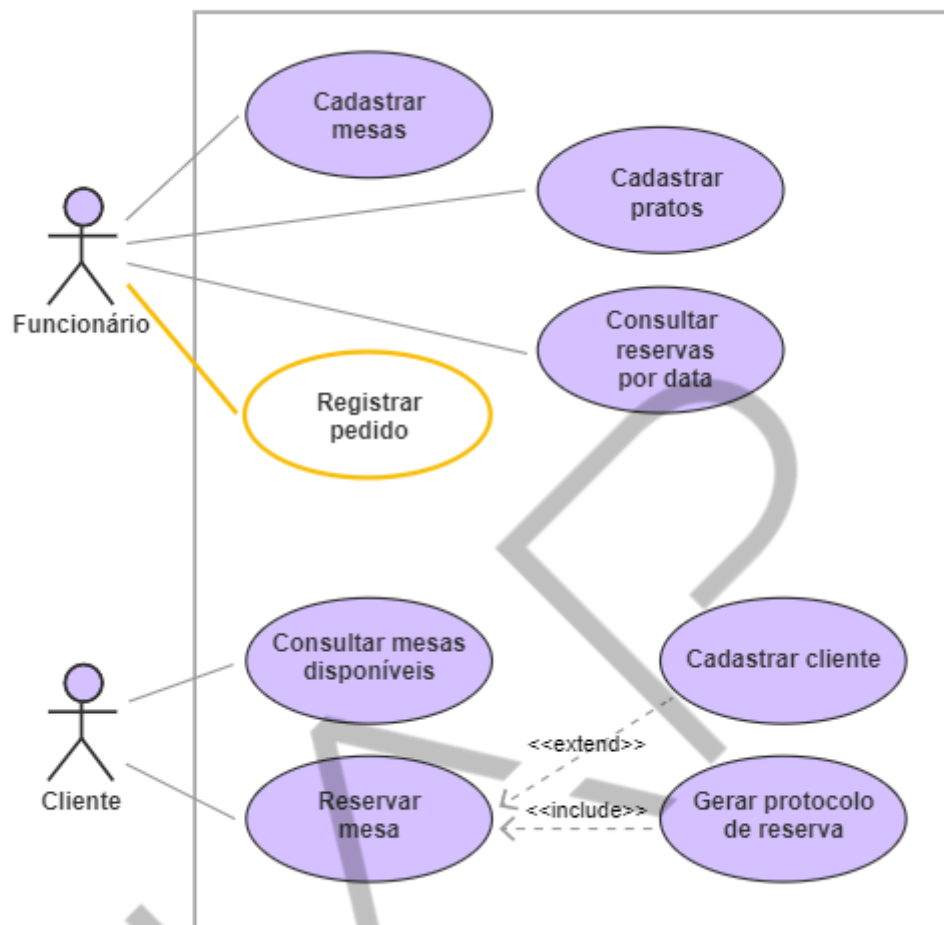


Figura 4.12 – Exemplo de Diagrama de Caso de Uso referente ao processo Reservar Mesa  
Fonte: Elaborado pela autora Cristina (2016)

#### 4.6.5 Diagrama de Classe

- **Objetivo:** é um diagrama estático que representa a estrutura lógica do sistema, subsistema ou componente projetado como classes e interfaces relacionadas, com suas características, restrições e associações, generalizações etc.
- **Visão Lógica ou Projeto:** descreve o sistema internamente, dando suporte à visão estrutural do sistema.

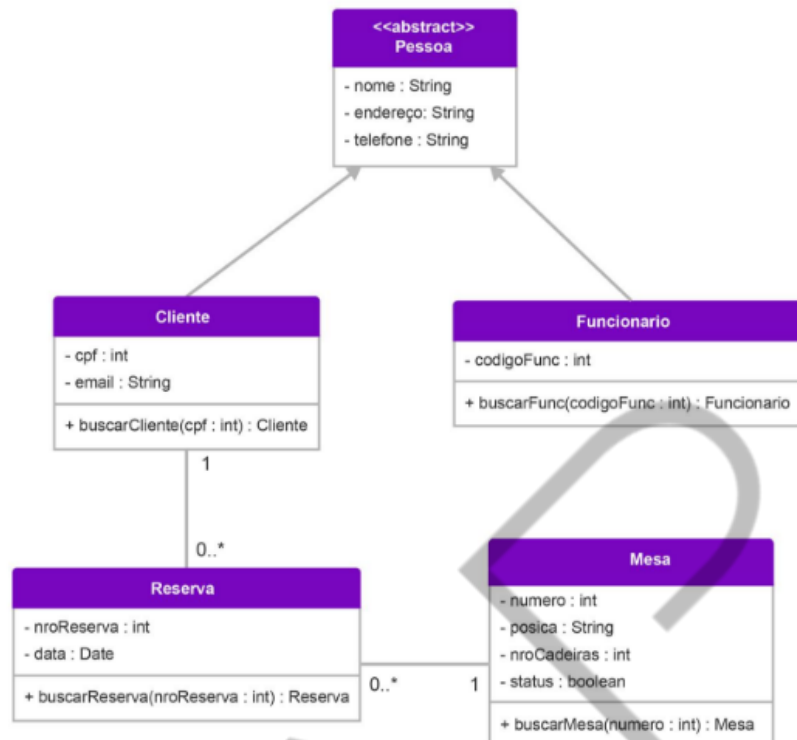


Figura 4.13 – Exemplo de Diagrama de Classe referente ao processo Reservar Mesa com notação de Pacotes  
 Fonte: Elaborado pela autora (2016)

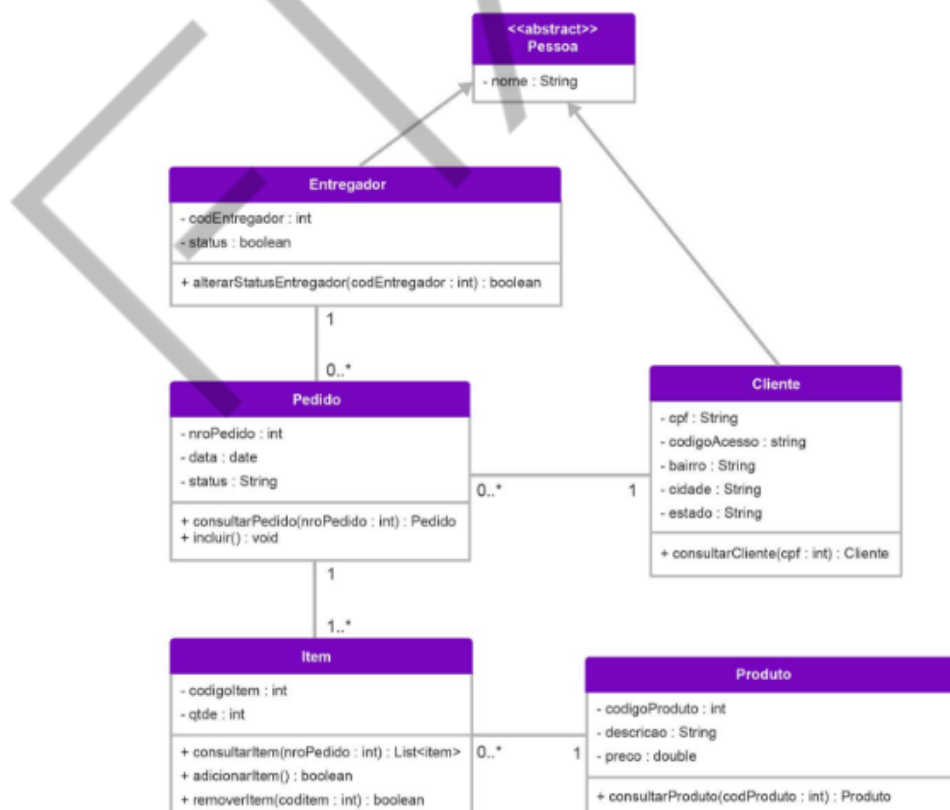


Figura 4.14 – Exemplo de Diagrama de Classe referente ao processo de Pedido  
 Fonte: Elaborado pela autora (2016)

#### 4.6.6 Diagrama de Sequência

- **Objetivo:** é um diagrama dinâmico que representa a ordem da troca de mensagens entre os objetos.
- **Visão de Concorrência:** trata da divisão do sistema em processos e processadores. Permite uma melhor utilização do ambiente onde o sistema se encontrará, se ele possui exceções paralelas e se existem gerenciamentos assíncronos.

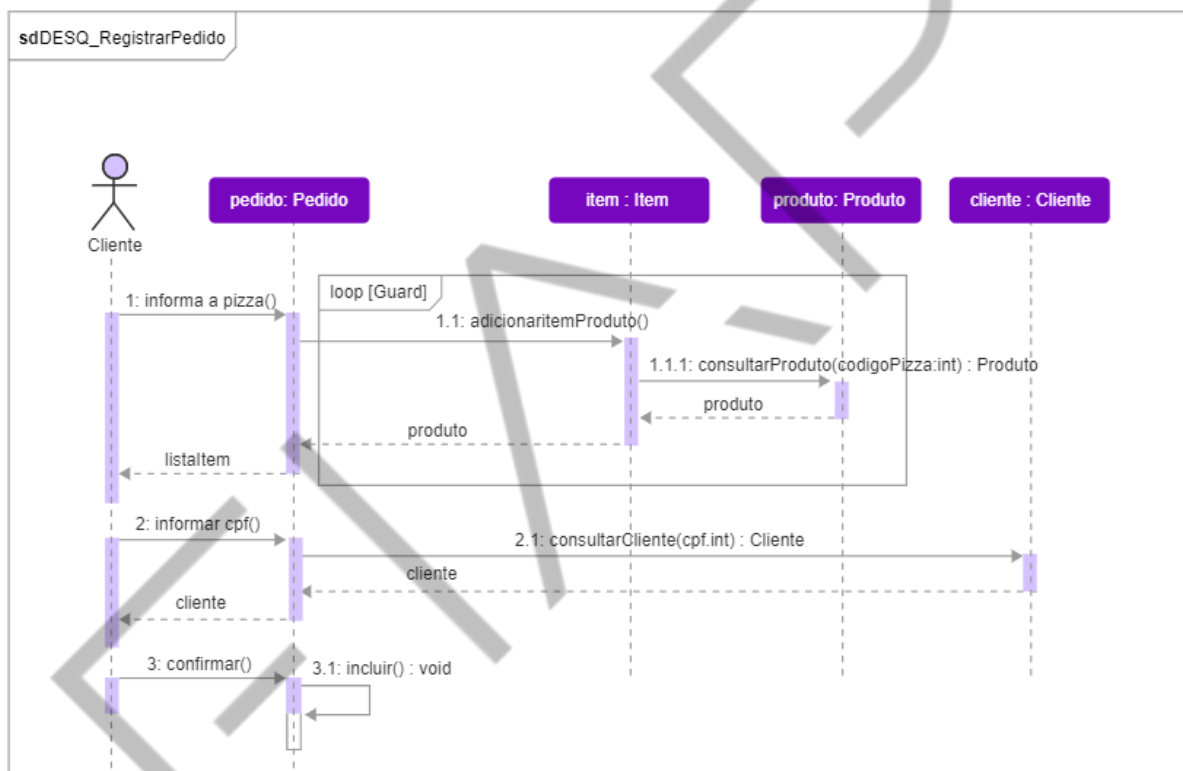


Figura 4.15 – Exemplo de Diagrama de Sequência de um processo de Pedido  
Fonte: Elaborado pela autora Cristina (2016)

#### 4.6.7 Diagrama de Máquina de Estados

- **Objetivo:** é um diagrama dinâmico que representa a situação em que um objeto se encontra em determinado momento durante o processamento. Um objeto pode passar por diversos estados.
- **Visão de Concorrência:** trata da divisão do sistema em processos e processadores. Permite uma melhor utilização do ambiente onde o sistema

se encontrará, se o mesmo possui exceções paralelas e se existem gerenciamentos assíncronos.

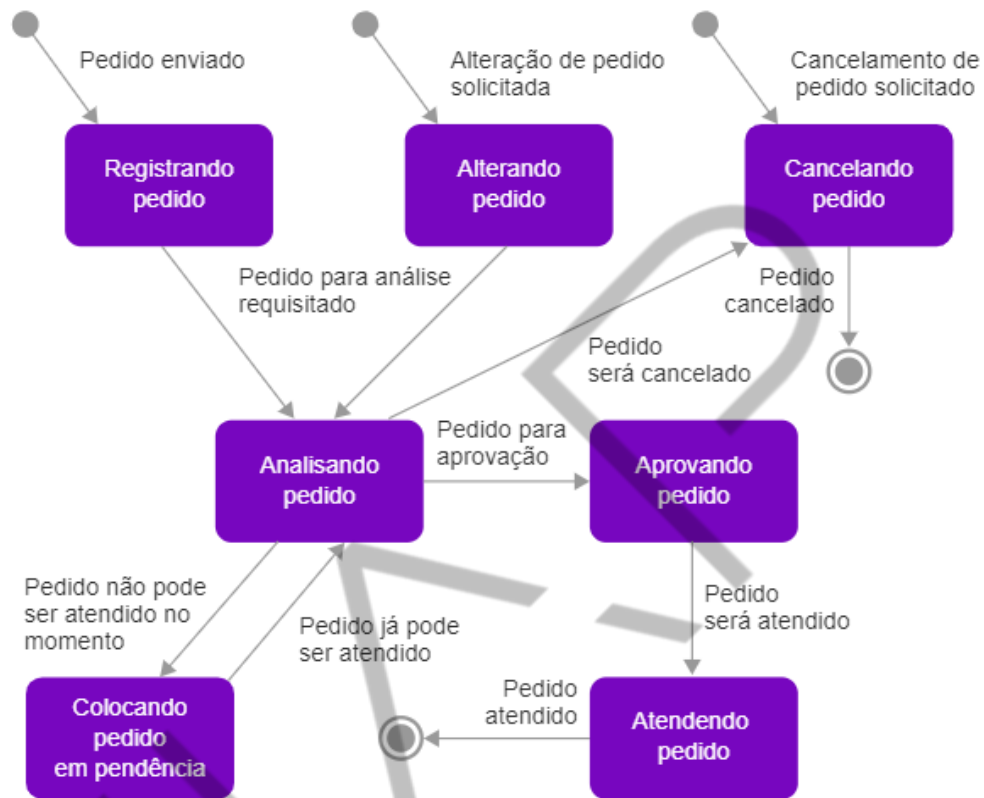


Figura 4.16 – Exemplo de Diagrama de Estado  
Fonte: Bezerra (2007)

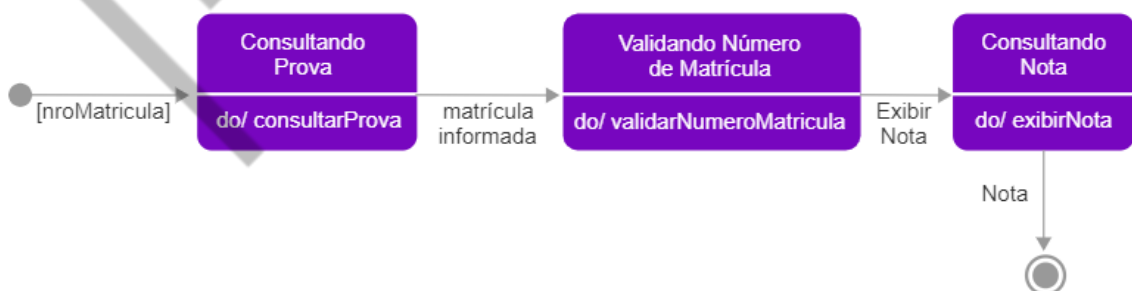


Figura 4.17 – Exemplo de Diagrama Máquina de Estados  
Fonte: Elaborado pela autora Cristina (2017)

#### 4.6.8 Diagrama de Componentes

- **Objetivo:** é um diagrama estático que representa a estrutura física da implementação, é construído como parte da especificação da arquitetura do software.
- **Visão de Componentes:** trata da descrição da implementação dos módulos e suas dependências. São desenvolvidos por desenvolvedores que têm maior experiência em programação ou por arquitetos de software.

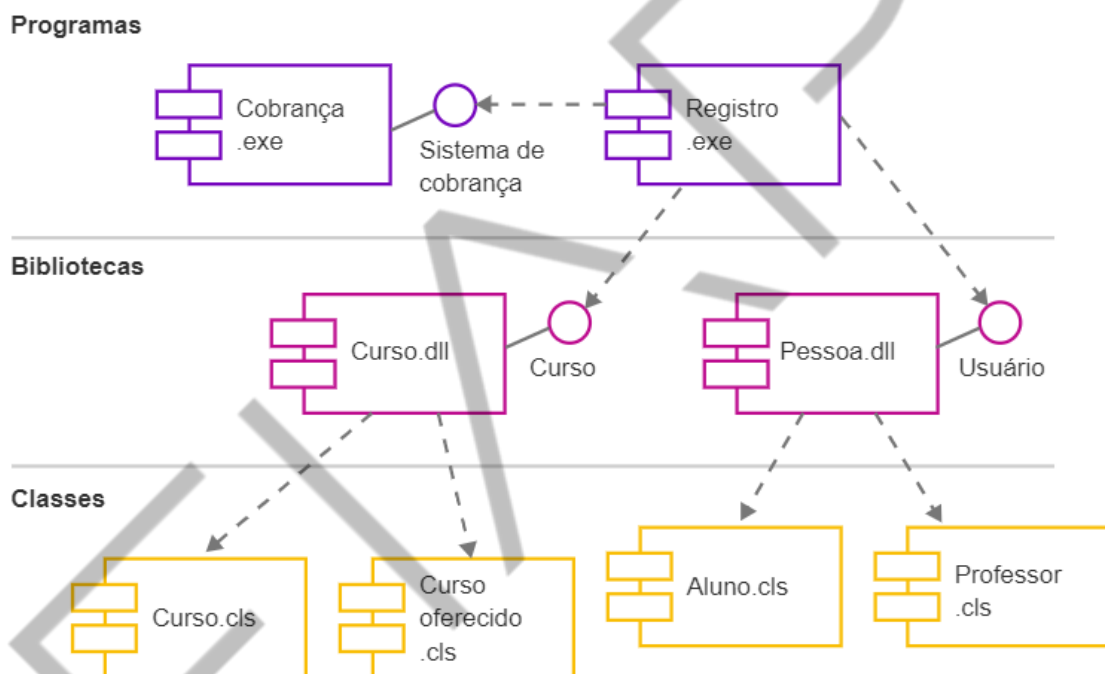


Figura 4.18 – Exemplo de Diagrama de Componentes, notação UML 1.4  
Fonte: Bezerra (2007)

A partir da UML 2, o componente que era representado por um retângulo com dois retângulos menores, conforme exemplo da Figura acima, foi substituído por um retângulo contendo internamente o antigo símbolo, conforme exemplo abaixo – Diagrama de Componente:

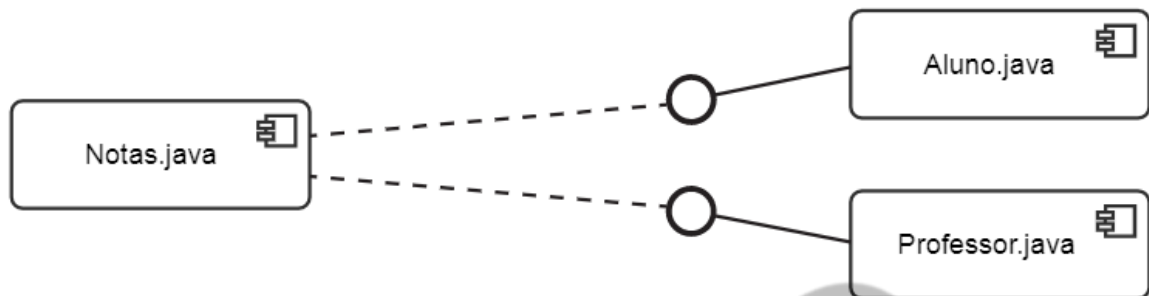


Figura 4.19 – Exemplo de Diagrama de Componente  
Fonte: Adaptado de Bezerra (2007)

#### 4.6.9 Diagrama de Implantação

- **Objetivo:** é um diagrama estático que representa os elementos de configuração do processamento em tempo de execução, ou seja, uma visão dos componentes de software.
- **Visão de Organização:** mostra a organização física do sistema, os computadores, os periféricos e como eles se conectam entre si.

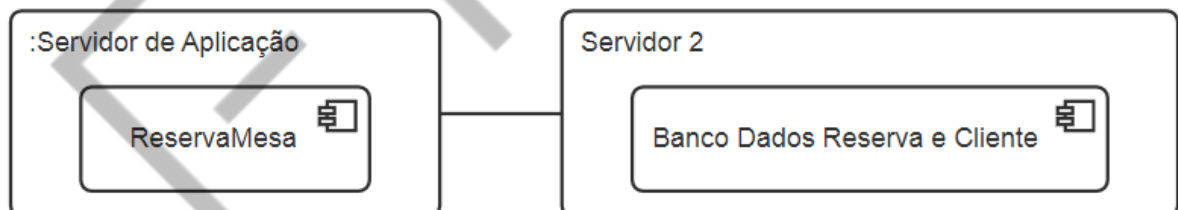


Figura 4.20 – Exemplo de Diagrama de Implantação com os componentes relacionados  
Fonte: Elaborado pela autora Cristina (2017)

#### 4.6.10 Diagrama de Objetos

- **Objetivo:** é um diagrama estático que representa a relação dos objetos com base nas instâncias criadas a partir do diagrama de classe de análise.

- **Visão Lógica ou Projeto:** descreve o sistema internamente, dando suporte à visão estrutural do projeto.

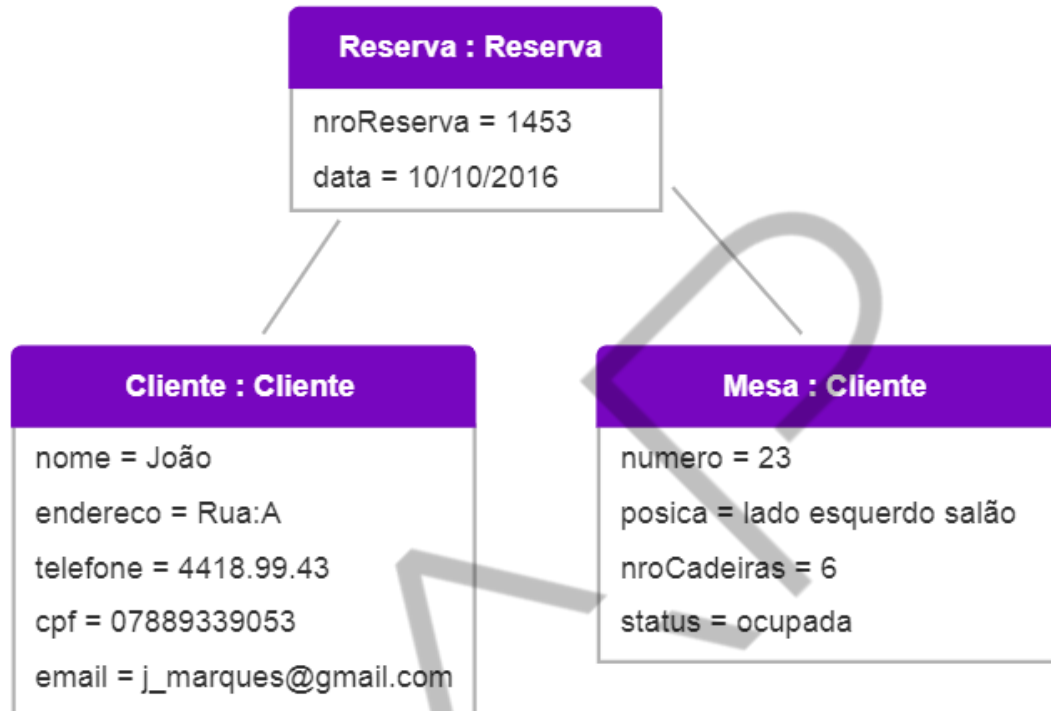


Figura 4.21 – Exemplo de Diagrama de Objetos  
Fonte: Elaborado pela autora Cristina (2017)

#### 4.6.11 Diagrama de Colaboração

- **Objetivo:** foco na ordenação estrutural onde as mensagens de um sistema são trocadas entre os objetos.
- **Visão Lógica ou Projeto:** este diagrama exibe de forma explícita a colaboração dinâmica entre os objetos.



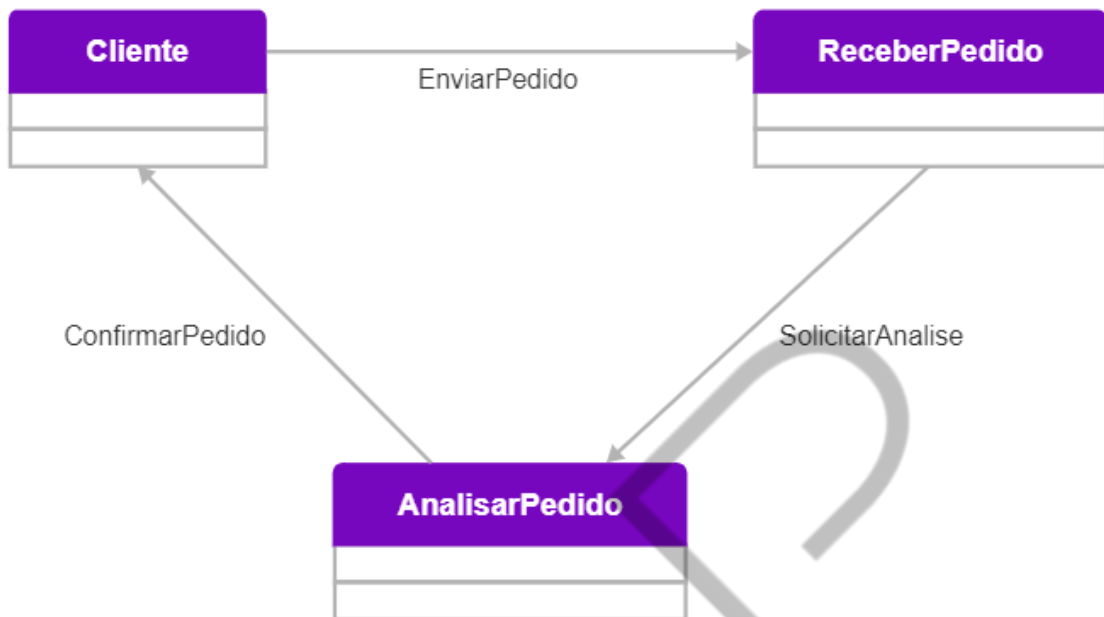


Figura 4.22 – Exemplo de Diagrama de Objetos  
Fonte: Wikipedia (2019)

#### 4.6.12 Diagrama de Pacotes

- **Objetivo:** ilustrar a arquitetura de um sistema mostrando o agrupamento de suas classes.
- **Visão Lógica ou Projeto:** descreve os pedaços e/ou pacotes do sistema divididos em agrupamentos lógicos.

#### 4.6.13 Diagrama de Perfil

- **Objetivo:** Definir novos elementos UML.
- **Visão Lógica ou Projeto:** estende diagramas existentes com a inclusão de estruturas customizadas.

No capítulo abordamos sobre:

- Diagrama de Atividades

- Diagrama de Caso de Uso
- Diagrama de Classe
- Diagrama de Sequência
- Diagrama de Máquina de Estados
- Diagrama de Componentes
- Diagrama de Implantação
- Diagrama de Objetos
- Colaboração
- Pacotes
- Perfil

## REFERÊNCIAS

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. 2. ed. São Paulo: Campus, 2007.

BOOCH, G.; RUMBAUGH, J.; JACBSON, I. **UML – Guia do Usuário**. 2. ed. São Paulo: Campus, 2006.

GUEDES, G. T. A. **UML 2 – Uma Abordagem Prática**. 2. ed. São Paulo: Novatec, 2011.

LARMAN, C. **Utilizando UML e Padrões**. 3. ed. Porto Alegre: Bookman, 2007.

OMG. **About the Unified Modeling Language Specification version 2.5.1**. Disponível em: <<https://www.omg.org/spec/UML/2.5.1/>>. Acesso em: 5 abr. 2019.

PFLEEGER, S. L. **Engenharia de Software – Teoria e Prática**. 2. ed. São Paulo: Prentice Hall, 2004.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 2011.

SBROCCO, J. H. T. C. **UML 2.3 – Teoria e Prática**. São Paulo: Érica, 2011.

SOMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.