RONNEESLEY MOURA TELES

PROJETO DE SISTEMAS USANDO UML

SUMÁRIO

1 INTRODUÇÃO

Este livro trata da modelagem de sistemas com base na notação *Unified Modeling Language* (UML) descrita por Booch, Rumbugh e Jacobson (2006). Atualmente a Orientação a Objetos é um dos paradigmas de programação mais utilizados no mercado, por isso sua importância acadêmica. Os pré-requisitos deste assunto são os conceitos de lógica de programação.

1.1 Ementa

Conceitos de análise orientada a objetos. Notação da Linguagem de modelagem unificada (UML). Fluxo de trabalho e resultado da fase de: análise de requisitos, análise e projeto do sistema.

1.2 Ferramentas utilizadas

- **Astah**: ferramenta utilizada para a criação de diagramas seguindo a notação UML. Pode ser obtida gratuitamente em ⟨http://astah.net⟩[1].
- **Netbeans**: Ambiente de Desenvolvimento Integrado (IDE) para programação em várias linguagens, como: Java, PHP, C e C++. Pode ser obtido gratuitamente em ⟨https://netbeans.org⟩.
- **LibreOffice**: ferramentas de escritório contendo editor de textos, planilhas, slides e outras. Pode ser obtido gratuitamente em ⟨https://pt-br.libreoffice.org⟩.

1.3 Exercícios

1. Qual a versão atual das ferramentas Astah, Netbeans e LibreOffice?
2. Quais os demais produtos além da ferramenta Astah Community? E o que eles fazem?
3. Cite pelo menos três alternativas ao Netbeans. Descreva características de cada uma delas.
4. Cite pelo menos três alternativas ao LibreOffice. Descreva vantagens e desvantagens de cada uma.

[1] Apenas a distribuição Community é gratuita.

5. Quaal a versão atual do UML? Quais as diferenças do UML 1.0 para o 2.0?

6. Descreva pelo menos dois outros paradigmas de programação.

7. O que faz a ferramenta Gliffy disponível em ⟨www.gliffy.com⟩?

2 ANÁLISE DE SISTEMAS

O analista de sistemas é um profissional capaz de concretizar as necessidades do usuário em um *software*. É ele a pessoa capaz de entender o que o usuário precisa e transformar estas necessidades em um produto, mesmo que o próprio usuário não saiba exatamente o que deseja.

2.1 A importância da criação de modelos

A história mostra que o homem deu valor a diversos tipos de bens: propriedade, mão de obra, máquinas e capital. A informação emergiu nesse cenário mostrando seu valor como um novo bem das empresas. A empresa com maior número de informações sobre o seu processo de negócio possui vantagem na competição pelo mercado (BEZERRA, 2014).

"A necessidade é a mãe das invenções". Com o crescimento da importância da informação, surge a necessidade de gerenciá-la de forma adequada e eficiente, assim surgiram os Sistemas de Informações. Um sistema de informações é a combinação de: pessoas, dados, processos, interfaces, redes de comunicação e tecnologia. Estes elementos interagem com o objetivo de melhorar o processo de negócio de uma empresa (BEZERRA, 2014).

"O objetivo principal e final da construção de um sistema de informações é a adição de valor à empresa ou organização na qual esse sistema será utilizado" (BEZERRA, 2014, p. 1). Adição de valor quer dizer que os processos da empresa devem ser melhorados para compensar a construção do sistema, ou seja, o sistema deve ser economicamente justificável. Desenvolver um sistema de informações é uma tarefa muito complexa (BEZERRA, 2014, p. 1).

Um dos componentes de um sistema de informações é o sistema de *software*. Este componente possui "módulos funcionais computadorizados que interagem entre si para proporcionar ao(s) usuário(s) do sistema a automatização de diversas tarefas" (BEZERRA, 2014, p. 2).

Uma característica dos sistemas de *software* é que a complexidade de seu desenvolvimento cresce proporcionalmente ao seu tamanho. Neste sentido, Bezerra (2014) compara os itens necessários para desenvolver uma casa. Se esta casa fosse para um cachorro, seriam necessárias ripas de madeira, pregos, caixa de ferramentas, amor pelo animal e alguns dias de trabalho. No entanto, se fosse uma casa para sua família a tarefa não seria tão fácil. O tempo e os recursos seriam algumas dezenas de vezes maiores. Já se a construção trate-se de um edifício certamente seria necessário um esforço ainda maior.

"Os engenheiros e arquitetos constroem plantas de diversos elementos da habitação antes do início da construção..." (BEZERRA, 2014, p. 2). As plantas hidráulicas, elétricas, de fundação e outras devem manter consistência entre si.

A construção de sistemas de *software* também ocorre este aumento da complexidade conforme o tamanho do sistema. "O equivalente ao projeto das plantas da engenharia civil também deve ser realizado" (BEZERRA, 2014, p. 2). Este é o conceito de modelo: "... um modelo pode ser visto como uma representação idealizada de um sistema a ser construído." (BEZERRA, 2014, p. 2). Por exemplo maquetes de edifícios, aviões, plantas de circuitos eletrônicos (BEZERRA, 2014, p. 2).

Segundo Bezerra (2014, p. 2 e 3) os modelos de sistemas são construídos para lidar com a complexidade que é a construção de um *software*. Um sistema pode ter vários modelos para a sua construção, no caso de *software* constrói-se geralmente o diagrama de casos de uso e o diagrama de classes.

Além disso, estes modelos permitem a comunicação entre as pessoas envolvidas, uma vez que um problema pode ser resolvido de diversas formas, os modelos especificam a forma escolhida pela equipe para a resolução. Cotidianamente pode-se usar os modelos como um contrato entre a empresa que fornece o *software* e o cliente que o solicita Bezerra (2014, p. 3).

Adicionalmente, os modelos permitem a correção de erros de comunicação e individuais antes mesmo deles ocorrerem. A correção no modelo custa menos tempo e recursos que a correção no *software*. Finalmente, estes modelos permitem a discussão do comportamento do sistema antes mesmo da construção Bezerra (2014, p. 3).

Estes modelos podem tanto serem representados por diagramas como também por textos. Estas informações textuais geralmente complementam dados apresentados nos diagramas. Neste sentido um diagrama juntamente aos seus textos associados formam a *documentação* do modelo Bezerra (2014, p. 4).

2.2 Especificação do sistema

A especificação do sistema "é um documento que serve de base para a engenharia de *hardware*, engenharia de *software*, engenharia de banco de dados e engenharia humana" (PRESSMAN, 1995). Após as primeiras reuniões com os usuários são definidos os requisitos do sistema (SOMMERVILLE, 2007, p. 79).

2.3 Os requisitos

Segundo Macoratti (2012) um requisito é uma função, restrição ou propriedade que deve ser fornecida pelo *software*. Os requisitos satisfazem as necessidades dos usuários. De forma geral: "descreve um serviço ou uma limitação". Devem ser definidos com objetividade e clareza, sendo que a mal definição dos requisitos juntamente as alterações constantes neles são as principais causas de fracasso dos projetos.

Não existe um modelo geral para descobrir os requisitos do sistema. A sugestão é descobri-los em consultas, documentos, pesquisas, entrevistas e JAD (*Joint Application Design)*, análise dos requisitos identificados, refinamento e detalhamento, modelagem e validação dos requisitos (verificação da consistência), e acompanhamento dos requisitos em todas as fases.

Os requisitos são classificados como funcionais e não funcionais. Estes serão descritos nas seções 2.3.2 e 2.3.3.

2.3.1 Metodologia JAD

Segundo Rocha (2017) a metodologia JAD foi criada pela International Business Machines (IBM). Baseia-se em reuniões com a filosofia de que todos os participantes são coautores da solução.

Nestas reuniões sugere-se dinâmicas de grupos conduzidas por um facilitador e com participação dos usuários chaves. Sempre quando possível sugere-se o uso de recursos audiovisuais, pois facilitam a comunicação e o entendimento (ROCHA, 2017).

Durante as discussões deve-se analisar todos os pontos do projeto com uma abordagem *top-down*. Isto quer dizer que o sistema será analisado do ponto mais amplo para os detalhes operacionais (ROCHA, 2017).

Em cada reunião recomenda-se a elaboração de um documento com as decisões tomadas. Este documento deverá ser assinado pelos participantes como uma prova de acordo entre as decisões realizadas (ROCHA, 2017).

Cada etapa do JAD possui fases: adaptação, sessão e finalização. Na adaptação planeja-se as reuniões e todo o material a ser utilizado e convida-se os participantes, a sessão por sua vez é a própria reunião, é nesta fase então que desenvolve-se e documenta-se os requisitos, e por último, a finalização que converte os requisitos em um documento chamado documento de especificação de requisitos (ROCHA, 2017).

2.3.2 Requisitos funcionais

Descrevem as funcionalidades do sistema desejadas pelos *stakeholders* (interessados). Assim, um requisito funcional determina um comportamento esperado do sistema (MACORATTI, 2012).

Sommerville (2007, p. 80) descreve os requisitos como serviços fornecidos pelo sistema e suas restrições operacionais. O processo que envolve a descoberta, analise, documentação e verificação destes requisitos é chamado de engenharia de requisitos. Os requisitos podem ser especificados com dois diferentes níveis de detalhes que são classificados como *requisitos de usuários* e *requisitos de sistema*.

Os requisitos de usuários possuem uma descrição abstrata do que o sistema deve fazer, evitando sempre que possível impor a solução para o problema a ser trabalhado no requisito. Estes requisitos são especificados pelo cliente que demanda a solução. Já os requisitos de sistema são especificados pelos fornecedores, neste determina-se com clareza e objetividade o que o sistema deve fazer. Este último pode ser parte do contrato estabelecido pelas partes (SOMMERVILLE, 2007, p. 80).

A seguir alguns exemplos: 1) O sistema deve fornecer um cadastro de clientes; 2) O sistema deve emitir um relatório de aniversariantes; 3) O sistema deve fornecer um cadastro de fornecedores. As tabelas 1, 2 e 3 sugerem modos de documentar os requisitos funcionais.

Nota-se que também é recomendável anotar informações a respeito do surgimento do requisito como sua fonte, data e local da reunião em que detectou-se o requisito. Estas informações complementam e auxiliam no rastreamento de informações.

Tabela 1 – Requisito Funcional (Exemplo R1): O sistema deve fornecer um cadastro de produtos.

Identificação do Requisito	**Exemplo R1**
Nome do Requisito	O sistema deve fornecer um cadastro de produtos
Fonte do Requisito	Sebastião
Data	8 de março de 2012
Local e/ou Reunião	Sala 10 da empresa A
Responsável pelo Requisito	Ronneesley Moura Teles
Especificação do Requisito	

Um produto possui os seguintes dados: código, nome, quantidade em estoque, descrição, preço de compra e preço de venda. A quantidade em estoque de um produto não pode ser negativa. Seu preço de compra deve ser um valor superior a R$ 0,00. O preço de venda do produto não pode ser inferior ao preço de compra. O nome do produto, quantidade, preço de compra e de venda devem ser obrigatoriamente informados.

Tabela 2 – Requisito Funcional (Exemplo R2): O sistema deve fornecer um cadastro de fornecedores.

Identificação do Requisito	**Exemplo R2**
Nome do Requisito	O sistema deve fornecer um cadastro de fornecedores
Fonte do Requisito	João
Data	13 de março de 2012
Local e/ou Reunião	Empresa X
Responsável pelo Requisito	Ronneesley Moura Teles
Especificação do Requisito	
Os fornecedores possui código, nome, CNPJ, telefone e e-mail, sendo que os campos nome e CNPJ são obrigatórios. Cada empresa pode fornecer um conjunto de diferentes produtos. O e-mail e CNPJ devem ser válidos.	

Tabela 3 – Requisito Funcional (Exemplo R3): O sistema deve emitir um relatório dos produtos em estoque.

Identificação do Requisito	**Exemplo R3**
Nome do Requisito	O sistema deve emitir um relatório dos produtos em estoque
Fonte do Requisito	Marcia
Data	13 de março de 2012
Local e/ou Reunião	Empresa Y

Responsável pelo Requisito	Ronneesley Moura Teles
Especificação do Requisito	
Devem aparecer no relatório: o código do produto, seu nome, quantidade, preço de venda e total (quantidade vezes preço de venda). O relatório deve ser emitido em formato A4 em orientação retrato.	

2.3.3 Requisitos não funcionais

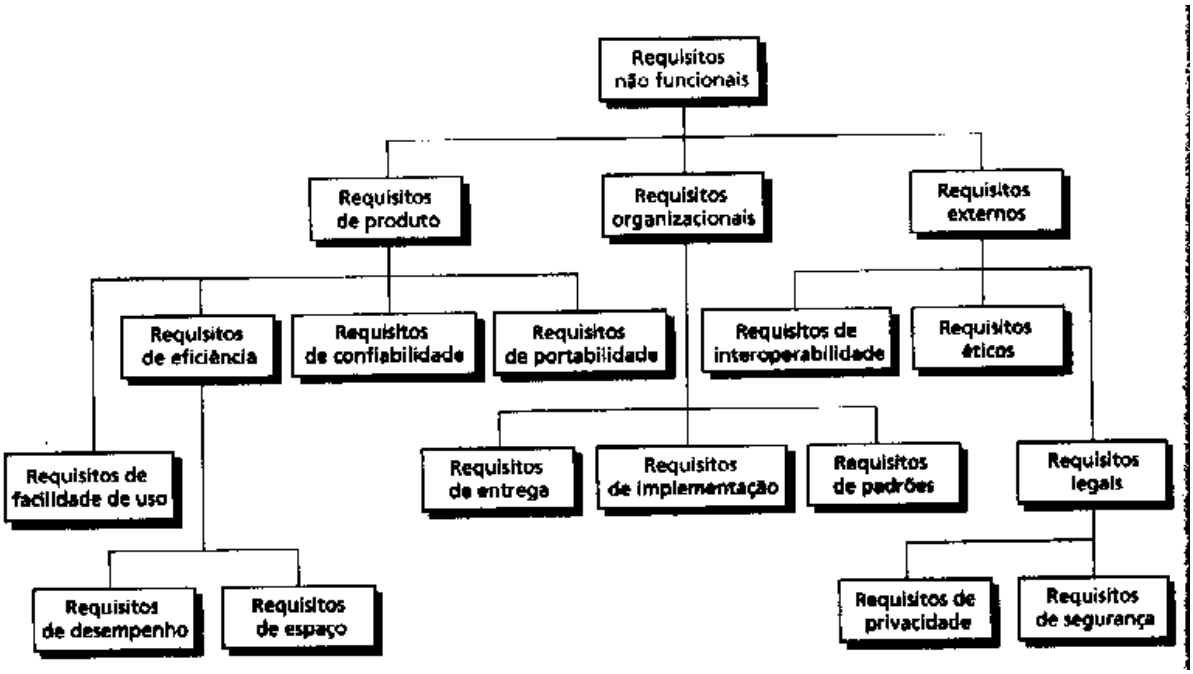
Segundo Macoratti (2012) os requisitos não funcionais são as "qualidades e restrições globais do sistema relacionados com: manutenção, uso, desempenho, custo, interface" e outros.

Sommerville (2007, p. 82) descreve os requisitos não funcionais como aqueles que não estão relacionados com as funções específicas do sistema, podendo estar relacionados com as propriedades emergentes do sistema como confiabilidade, tempo de resposta, espaço de armazenamento, desempenho, proteção e disponibilidade. Consideram-se propriedades emergentes aqueles que se manifestam em relação ao conjunto de todas as partes do sistema.

Deste ponto de vista os requisitos não funcionais são geralmente mais importantes que os funcionais, uma vez que a falha em um destes requisitos pode significar que o sistema todo é inútil (SOMMERVILLE, 2007, p. 82).

Ainda neste sentido estes requisitos podem até especificar o processo pelo qual o sistema será desenvolvido, podendo especificar por exemplo as ferramentas que serão utilizadas para tal ou padrões de qualidade de processo. Outra natureza para eles está na definição do orçamento, necessidade de interoperabilidade com outros sistemas (*software* ou *hardware*), fatores como legislações ou normas de segurança (SOMMERVILLE, 2007, p. 82). A Figura 1 apresenta os tipos de requistos não funcionais em um *software*.

Figura 1 – Tipos de requisitos não funcionais em um *software*. Fonte: (SOMMERVILLE, 2007, p. 82).



As tabelas 4, 5, 6 e 7 apresentam exemplos de requisitos não funcionais e sugerem uma forma de documentá-los.

Tabela 4 – Requisito Não Funcional (Exemplo RNF1): O SGBD deve ser protegido com usuário e senha.

Identificação do Requisito	**Exemplo RNF1**
Nome do Requisito	O SGBD deve ser protegido com usuário e senha
Fonte do Requisito	David
Data	13 de março de 2012
Local e/ou Reunião	Empresa X
Responsável pelo Requisito	Ronneesley Moura Teles
Especificação do Requisito	
Não deve haver nenhum usuário sem senha e todas elas devem ser compostas de no mínimo 7 caracteres, incluindo números, letras e caracteres especiais. Cada usuário do sistema possui um usuário próprio no SGBD.	

Tabela 5 – Requisito Não Funcional (Exemplo RNF2): O software deve fucionar no sistema operacional Window e Linux.

Identificação do Requisito	**Exemplo RNF2**
Nome do Requisito	O software deve fucionar no sistema operacional Window e Linux
Fonte do Requisito	João
Data	13 de março de 2012
Local e/ou Reunião	Centro de Reuniões
Responsável pelo Requisito	Ronneesley Moura Teles
Especificação do Requisito	
Todos os componentes do software devem funcionar no sistema operacional Windows e Linux.	

Tabela 6 – Requisito Não Funcional (Exemplo RNF3): O software deve consumir no máximo 1GB de RAM, ocupar no máximo 2GB de espaço e funcionar com um CPU de 1GHz.

Identificação do Requisito	**Exemplo RNF3**
Nome do Requisito	O software deve consumir no máximo 1GB de RAM, ocupar no máximo 2GB de espaço e funcionar com um CPU de 1GHz
Fonte do Requisito	João
Data	13 de março de 2012
Local e/ou Reunião	Centro de Reuniões
Responsável pelo Requisito	Ronneesley Moura Teles
Especificação do Requisito	
As respostas do software não podem exceder o tempo de 1 minuto.	

Tabela 7 – Requisito Não Funcional (Exemplo RNF4): O tempo de desenvolvimento do software não pode ultrapassar do dia 15/06/2017.

Identificação do Requisito	**Exemplo RNF4**
Nome do Requisito	O tempo de desenvolvimento do software não pode ultrapassar do dia 15/06/2017
Fonte do Requisito	João
Data	10 de fevereiro de 2017
Local e/ou Reunião	Centro de Reuniões
Responsável pelo Requisito	Ronneesley Moura Teles
Especificação do Requisito	
Se o prazo não for executado no tempo previsto haverá multa e o sistema poderá ser rejeitado pelo contratante, uma vez que será utilizado em uma situação promocional da empresa X durante uma data festiva.	

2.4 Documento de requisitos de software

O documento de requisitos de software também chamado de especificação de requisitos de software (SRS)[1] documenta o que os desenvolvedores deverão fazer. Deve incluir os requisitos de usuários e requisitos de sistemas. Caso a quantidade de páginas dos requisitos de sistemas seja muito grante, este pode ser apresentado em um documento separado (SOMMERVILLE, 2007, p. 91).

O detalhamento utilizado na SRS depende do tipo de sistema e do processo de desenvolvimento adotado. Quando o sistema for desenvolvido por um fornecedor externo as especificações devem ser precisas e detalhadas. No entanto, quando o processo for flexível, interno e interativo o documento pode ser menos detalhado, assim as ambiguidades são resolvidas durante o desenvolvimento (SOMMERVILLE, 2007, p. 91).

A organização mais conhecida deste documento é especificada pela IEEE/ANSI 830-1998. Este padrão sugere a estrutura do documento de uma forma ampla. Sommerville (2007, p. 92 e 93) expande este padrão com informações úteis a serem adicionadas.

[1] O significado da sigla é Software Requeriments Specification.

1. Prefácio: define o objetivo do documento, o público-alvo e descreve o histórico de versões, assim como uma justificativa para a mudança de versão;
2. Índices: podem ser incluídos diversos tipos de índices, como: sumário, lista de figuras, lista de tabela, etc. A medida que o documento se torna extenso os índices são mais necessários, pois permitem encontrar as informações.
3. Introdução: descreve a necessidade do sistema, descreve brevemente suas funções e interconexões com outros sistemas.
    3.1. Escopo do produto: nome, objetivos, limites de atuação, benefícios.
    3.2. Glossário: apresenta todas as definições, os acrônimos, as abreviaturas e o jargão da área
    3.3. Referências
    3.4. Visão geral do restante do documento: resumo breve das seções do documento
4. Descrição geral
    4.1. Perspectiva do produto: descrever situação ambiental de uso do *software*, interconexões, interfaces externas, operações, restrições de memória primária e secundária;
    4.2. Funções do produto: resumir como as funções colaboram para atingir os objetivos do *software*.
    4.3. Características dos usuários: perfil de conhecimentos esperados dos usuários;
    4.4. Restrições gerais: justificar as restrições tecnológicas de desenvolvimento;
    4.5. Suposições e dependências: itens ambientais que são externos ao sistema, como a disponibilidade de um SGBD previamente instalado.
5. Requisitos específicos: abrange todos os requisitos (funcionais e não funcionais), sendo a maior parte de todo o documento.
    a) Requisitos de usuário: deve usar ilustrações e todos os recursos para facilitar a comunicação de forma clara.
    b) Arquitetura do sistema: apresenta uma visão geral de alto nível da arquitetura do sistema, apresentando a distribuição das funções nos módulos.

c) Requisitos de sistema: descreve os requisitos funcionais e não funcionais.

6. Modelos de sistema: apresenta todos os modelos/diagramas desenvolvidos para o sistema, mostrando o relacionamento entre os componentes, sistema e ambiente.

7. Evolução de sistema: descreve mudanças previstas de *hardware*, necessidades do usuário, tecnológicas, etc.

8. Apêndices

9. Índice

O documento de especificação de requisitos é indispensável quando se trata de um fornecedor externo. Alguns métodos ágeis argumentam que os requisitos mudam tão rapidamente que o documento fica desatualizado, realizando assim um esforço desperdiçado (SOMMERVILLE, 2007, p. 93 e 94).

2.5 Exercícios

1. Cite três características importantes de um analista de sistemas.

2. O que são modelos? Cite pelo menos dois modelos que conhece de qualquer área.

3. O que é a relação "ganha-ganha"? Quais outros tipos de relações existem?

4. Cite dois diagramas desenvolvidos para construção de *software*.

5. O que são requisitos?

6. Quem criou a metodologia JAD? Como acontece o levantamento de requisitos nesta metodologia?

7. Quais as três fases da metodologia JAD? O que acontece em cada uma delas?

8. O que são requisitos funcionais?

9. O que são *stakeholders*?

10. Qual a diferença entre requisitos de usuários e requisitos de sistema? Quem os descreve?

11. O que são requisitos não funcionais? O que os diferencia dos requisitos funcionais?

12. Cite pelo menos três naturezas de requisitos não funcionais.

13. Elabore um requisito funcional e um não funcional para um sistema de bibliotecas.

14. Cite pelo menos dois métodos ágeis.

15. O que é um processo de desenvolvimento iterativo e incremental?

16. Você acha praticável desenvolver um *software* sem a escrita da SRS? Justifique.

17. Descreva o processo de desenvolvimento Scrum e Extreme Programming.

3 UML (UNIFIED MODELING LANGUAGE)

Segundo Booch, Rumbugh e Jacobson (2006) a UML é um linguagem de modelagem de sistemas. Ela não é uma metodologia de desenvolvimento, desta forma, não especifica o que deve ser feito, mas auxilia o projeto. Seus principais objetivos são: especificação, documentação, estruturação e visualização do funcionamento do sistema.

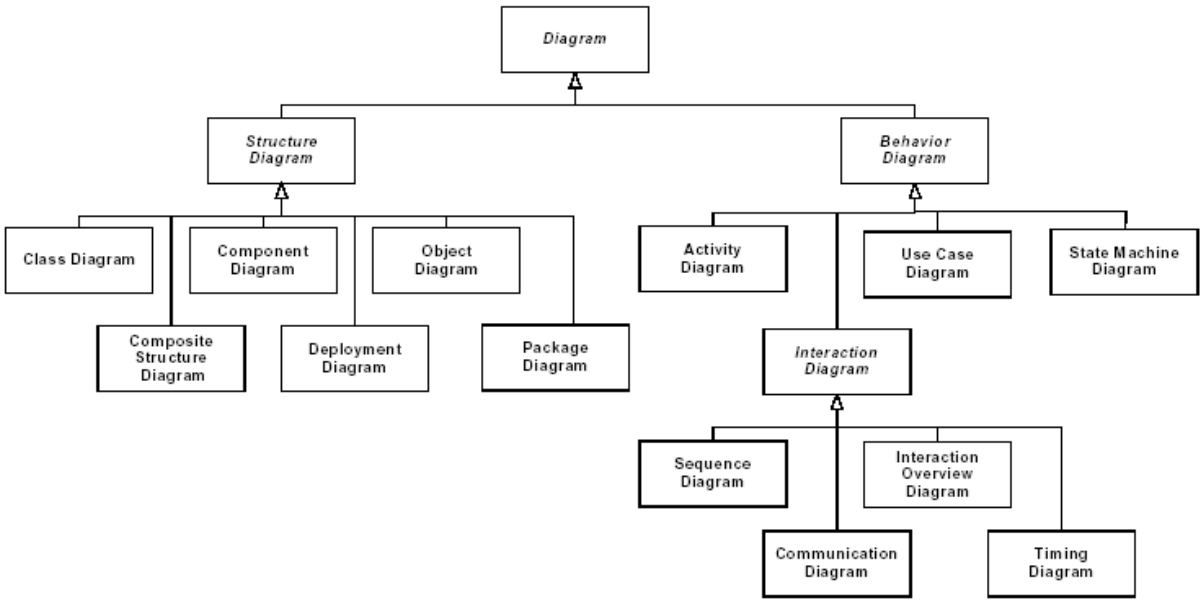
Sua base está na orientação a objetos. É originada das melhores práticas de projeto que provaram sucesso em sistemas grandes e complexos.

Deve-se usar a UML para documentar e conseguir entender de forma geral as necessidades do sistema e seus interessados (*stakeholders*); planejar com atencedência quais artefatos devem ser criados no projeto; saber quais classes, objetos, arquivos, biblioteca, outros artefatos serão criados ou serão incluídos no projeto; saber como os objetos se relacionaram entre si; saber quais recursos seram usados na implantação.

3.1 Diagramas da UML

A Figura 2 apresenta os principais diagramas da UML. Percebe-se que os diagramas são agrupados em: estruturais e comportamentais. Entre os diagramas estruturais pode-se destacar: o diagrama de classes, o diagrama de implantação e o diagrama de pacotes. Já para os diagramas comportamentais pode-se destacar: o diagrama de atividade, o diagrama de casos de uso e o diagrama de sequência.

Figura 2 – Organização dos diagramas da UML.



3.2 Exercícios

1. Qual o significado da sigla UML? Qual a versão atual da UML?
2. O que é um Diagrama de Fluxo de Dados (DFD)? Qual sua relação com a UML?
3. Quando surgiu a primeira versão da UML? Qual o número da sua versão?
4. Qual o paradigma de programação utilizado na UML?
5. E quais os objetivos da UML?
6. Quais os dois grupos de digramas da UML?
7. Existe algum vínculo da linguagem de programação escolhida com a UML? Pode-se escolher qualquer linguagem de programação?
8. Busque a Internet um diagrama de casos de uso e um diagrama de classes.

4 CASO DE USO

Em todo sistema ocorrem interações com atores humanos ou autômatos. Estes atores utilizam o sistema com algum propósito e esperam que o sistema se comporte como o previsto.

Um caso de uso especifica o comportamento do sistema ou de uma parte dele. É uma descrição de um conjunto de ações, tanto do ator quanto do sistema, incluindo variações no comportamento para produção do resultado desejado.

Os casos de uso podem especificar o comportamento pretendido pelo sistema que será desenvolvido, sem que se implemente códigos para isso. Eles fornecem uma compreensão comum do comportamento do sistema entre os desenvolvedores e as partes interessadas.

Neste sentido os casos de uso permitem a validação da arquitetura e verificação do sistema ao longo de sua evolução. Casos de uso bem-estruturados representam os comportamentos essenciais do sistema ou subsistema, não são amplamente gerais e nem específicos. Sua especificação depende do bom senso.

4.1 O Comportamento

Um *software* é muito mais que várias linhas de código, pois existe toda a documentação envolvida em sua construção. Na concepção desta o projetista pensa como será o uso do sistema refletindo sobre o modo como os atores irão interagir com o sistema e quais principais funcionalidades que utilizarão. Em seguida deve-se analisar as funcionalidades que auxiliam as principais elencadas no primeiro passo.

Após a enumeração de todas as funcionalidades, tanto as principais como as auxiliares, o projetista pensará nos detalhes de funcionamento das mesmas, especificando passo-a-passo as rotinas internas e suas relações com as demais funcionalidades. Vale ressaltar que este passo-a-passo não pode ser extremamente geral e nem específico demais. Deve-se ter em mente quesitos de qualidade de *software* como eficiência e facilidade de uso durante a construção destes passos.

Nesta modelagem são considerados os casos de uso do sistema que irão nortear a arquitetura do mesmo. A principal vantagem da utilização desta abordagem é a preocupação no que tem de ser feito, em vez de se preocupar em como fazer.

Seguindo estes princípios pode-se dizer que os casos de uso especificam o comportamento desejado, sem determinar como este comportamento será implementado. Desta forma

a comunicação entre o projetista, as partes interessadas (o usuário final e especialistas) e o programador (quem irá construir o sistema) acontece de forma mais natural sem a necessidade de muitos detalhes do funcionamento interno.

Os detalhes do funcionamento de um caso de uso apareceram na documentação. Esta forma de visualização é preferencial quando envolve as partes interessadas. Desta forma, um caso de uso é um conjunto de sequência de ações, incluindo suas variações de comportamento para se chegar a um resultado.

A nível de sistema o caso de uso é uma sequência de ações entre o ator e o próprio sistema. Esses comportamentos são utilizados para visualizar, especificar, construir e documentar o comportamento desejado do sistema durante a fase de análise de requisitos. Assim um caso de uso representa um requisito funcional como um todo.

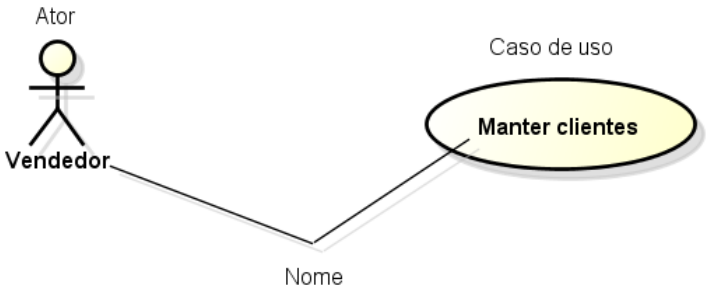
Um caso de uso documenta as interações do ator com o sistema. Este ator representa um conjunto coerente de papéis. Os atores podem ser humanos ou sistemas automatizados, como um robô ou um programa. Exemplos de atores em um sistema de controle de estoque são: a pessoa que trabalha no caixa, o administrador, o cliente (que acessa sua conta pela Internet), o fornecedor (responsável por atualizar seus preços de produtos ou serviços).

Um caso de uso pode ter variantes, ou seja, dentro dele podem ser utilizadas versões específicas de outro caso de uso. Por exemplo: quando um cliente se cadastra pode ser enviado a ele um e-mail e quando ele esquece sua senha, também é enviado um e-mail com a nova senha. Em ambos os casos um e-mail é enviado, este é o caso de uso específico utilizado no cadastro de clientes e na recuperação de senha.

Um caso de uso deve executar uma quantidade tangível de trabalho, devendo realizar algo que tem valor para o ator. Por exemplo: emissão de um relatório, cadastro de um cliente, alteração do endereço de um cliente.

A UML fornece uma representação gráfica de um caso de uso e de um ator. A Figura 3 apresenta esta representação gráfica.

Figura 3 – Representação gráfica de um caso de uso e de um ator.



4.2 Conceitos Importantes

Como citado, um caso de uso é um conjunto de sequência de ações, juntamente com as variantes que o sistema executa para produzir um resultado desejável pelo ator que o usa. A representação simbólica de um caso de uso na UML é uma elipse.

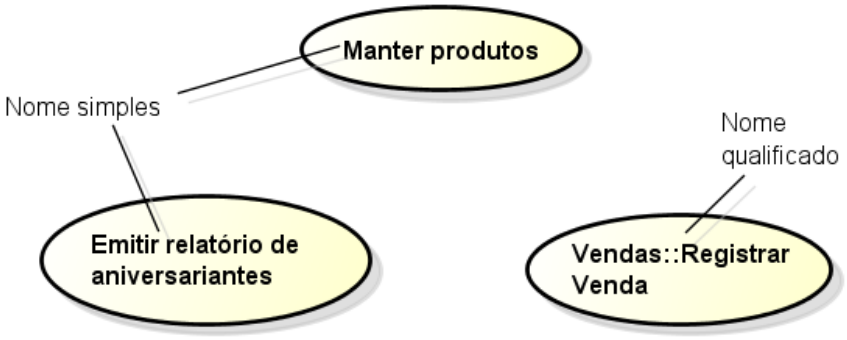
4.2.1 Assunto

Chama-se de **assunto**, uma classe descrita por vários casos de uso correlacionados. Normalmente um assunto é um sistema ou um subsistema. Sendo assim os casos de uso representam o comportamento de um assunto.

4.2.2 Nomenclatura

Todo caso de uso tem um nome que o identifica e é único dentro do pacote no qual ele se encontra. Quando esse nome aparece sozinho é chamado de **nome simples**. Normalmente o caso de uso é exibido somente com o nome simples. No entanto, em algumas situações o **nome de caminho/qualificado** é exibido. Chama-se de **nome de caminho** o nome do caso de uso com um prefixo que se refere ao pacote no qual ele se encontra. A Figura 4 apresenta um exemplo de nomes simples e qualificados (BOOCH; RUMBUGH; JACOBSON, 2006, p. 230).

Figura 4 – Casos de uso com nomes simples e com nomes qualificados.



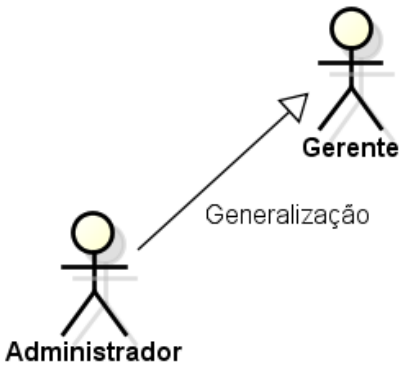
O nome do caso de uso é um texto podendo conter letra, números e a maioria dos sinais de pontuação, exceto: dois-pontos (pois é utilizado para identificar seu pacote). É importante ressaltar que o nome pode possuir várias linhas. Na prática utiliza-se nomes breves e que utilizem verbos para dar sentido se ação, que mapeie corretamente um comportamento do sistema.

4.2.3 Atores

Um ator representa um conjunto de papéis dentro do sistema. Normalmente um ator é uma pessoa, um *hardware* ou um programa. Embora os atores sejam representados graficamente nos diagramas, eles localizam-se fora do sistema. Desta forma, durante a execução do sistema, os atores não precisam existir como uma entidade separadas.

A Figura 5 apresenta a representação gráfica de um ator. uma generalização representa o ator mais geral e o mais específico. O ator mais específico está ligado ao triângulo da generalização (BOOCH; RUMBUGH; JACOBSON, 2006, p. 231).

Figura 5 – Generalização entre atores.



Os atores podem estar ligados aos casos de uso somente pelas associações. Nestes casos, a associação indica que o ator e o caso de uso se comunicam, ou seja, interagem um com o outro.

4.3 O Diagrama de Casos de Uso

O Diagrama de Casos de Uso tratam dos aspectos dinâmicos do sistema. Na UML os diagramas de atividades, gráfico de estados, sequência e comunicação também tratam de aspectos dinâmicos. Neste sentido este diagrama trata da modelagem de contexto do sistema, subsistema ou classe ou a modelagem dos requisitos do comportamento desse elementos. Ele apresenta uma visão externa sobre como os elementos podem ser utilizados (BOOCH; RUMBUGH; JACOBSON, 2006, p. 241).

Este digrama facilita o entendimento das funções do sistema para os usuários, uma vez que apresenta todas suas funcionalidades de forma fácil de entender. Para os desenvolvedores do sistema este diagrama também facilita a visão geral do comportamento do sistema e suas partes. Nele é apresentado um conjunto de casos, atores e seus relacionamentos (BOOCH; RUMBUGH; JACOBSON, 2006, p. 242 e 243).

Os elementos gráficos utilizandos nestes diagramas são: assunto, casos de uso, atores e relacionamentos (dependência, generalização e associação). Existem duas maneiras de especificar um determinado assunto através dos casos de uso: modelando o contexto de um assunto ou modelando os requisitos de um sistema (BOOCH; RUMBUGH; JACOBSON, 2006, p. 243 e 244).

4.3.1 Modelagem do Contexto de um Assunto

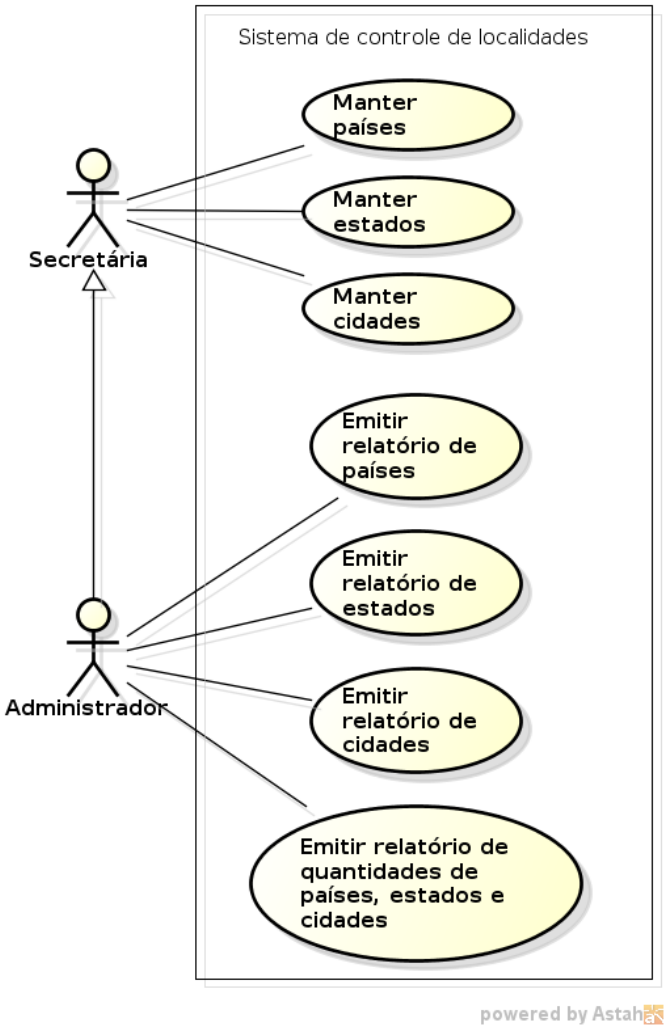
Nesta técnica determina-se quais atores estão dentro e quais estão fora do assunto e como eles interagem. Desta maneira o caso de uso especifica os atores e seus respectivos papeis. Tudo que está dentro do assunto é responsável pela execução do comportamento enquanto o que está fora é consumidor destes comportamentos. Tudo que está do lado de fora e interage com o sistema é chamado de **contexto do sistema**. É este contexto que define o ambiente no qual o sistema existe. A seguir os passos para se realizar este tipo de modelagem (BOOCH; RUMBUGH; JACOBSON, 2006, p. 244 e 245):

1. Identificar os limites (escopo) do sistema, determinando quais comportamentos são par-

tes dele e quais não são. Este limite é chamado de assunto;

2. Identificar os atores que se encontram ao redor do sistema, identificando quais grupos realizam funções secundárias de administração e manutenção;
3. Identificar a hierarquia dos atores através de generalizações/especializações;
4. Determinar um esteriótipo para cada ator para ajudar a compreensão.

Figura 6 – Diagrama de casos de uso modelado pelo contexto do assunto.



A Figura 6 apresenta um caso de uso modelado pelo contexto do assunto para um sistema de controle de localidades. Este diagrama apresenta dois atores e sete casos de usos ligados entre si por associações. O ator administrador é uma especialidade do ator secretária, especificado através de uma generalização.

4.3.2 Modelagem dos Requisitos de um Sistema

Nesta técnica especifica-se o que o assunto deve fazer sob um ponto de vista externo sem entrar em detalhes de como fazer. Desta forma o diagrama especifica o comportamento desejado do assunto (BOOCH; RUMBUGH; JACOBSON, 2006, p. 244).

Entende-se por requisito uma característica de projeto, uma propriedade ou um comportamento do sistema. Eles funcionam como um contrato estabelecido entre os elementos externos e internos do sistema. Chama-se se **sistema bem-comportado** aquele que executa seus requisitos de maneira fiel, previsível e confiável. Um requisito pode ser especificado de várias formas, desde um texto não-estruturado a expressões em uma linguagem formal (matemática). A maioria dos requisitos funcionais de um sistema podem ser expressos através de casos de uso e diagramas de casos de uso. A seguir os passos para se realizar a modelagem de requisitos de um sistema (BOOCH; RUMBUGH; JACOBSON, 2006, p. 246 e 247):

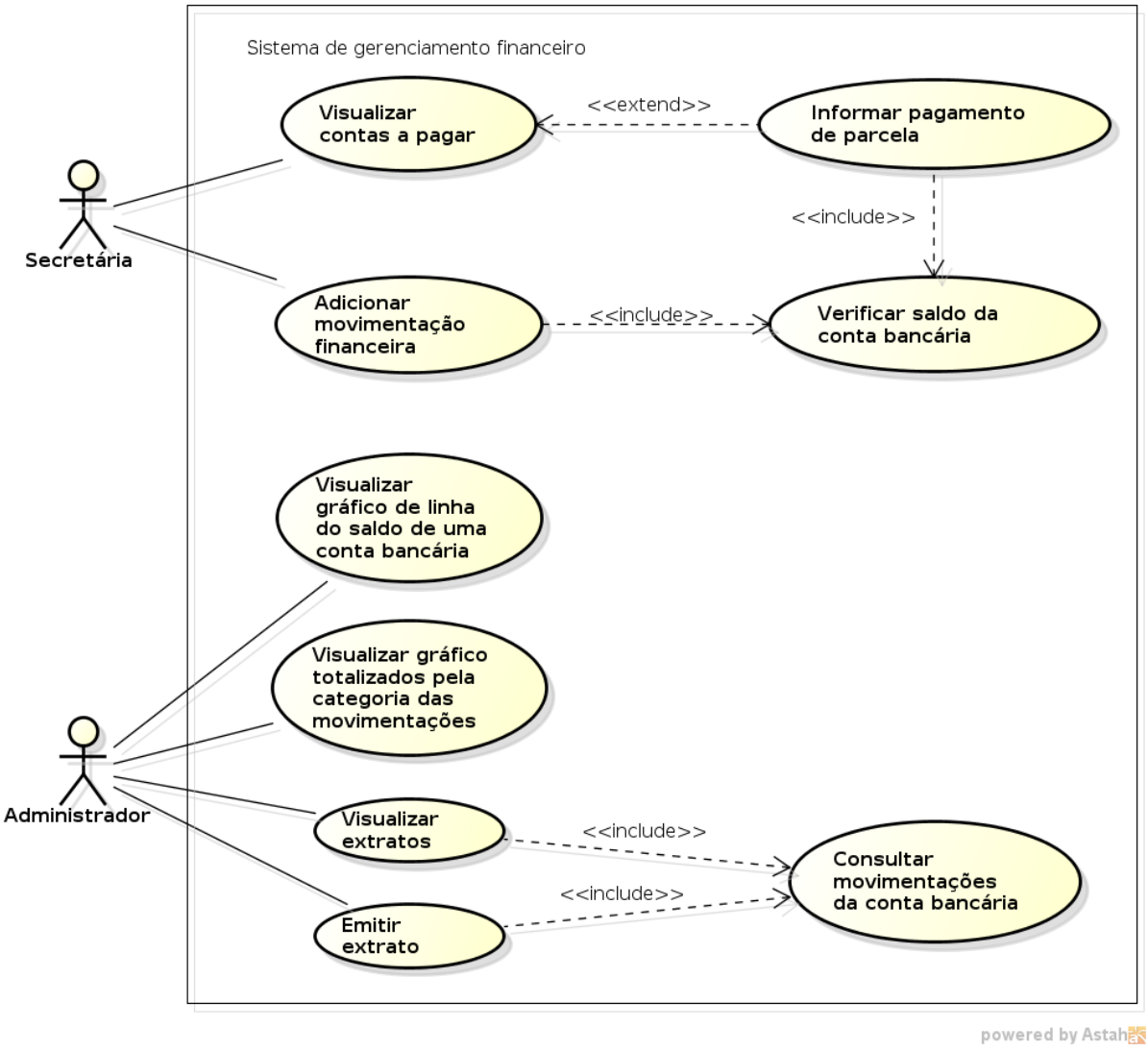
1. Identificar o contexto do sistema e especificar os atores ao redor dele;
2. Identificar para cada ator o comportamento esperado para o sistema;
3. Nomear os comportamentos como casos de uso;
4. Fazer a fatoração dos comportamentos comuns em novos casos de uso. Fazer extensões para os comportamento variante;
5. Modelar estes casos de uso, atores e relacionamentos em um diagrama de casos de uso;
6. Incluir adornos nos casos de uso quando se tratarem de requisitos não-funcionais.

A Figura 7 apresenta um diagrama de casos de uso modelado pelos requisitos do sistema. Neste caso existe uma diferença importante que é a informação de detalhes dos casos de uso, como acontece no caso de uso "Verificar saldo da conta bancária" que é utilizado pelos casos de uso "Informar pagamento de parcela" e "Adicionar movimentação financeira", e também no caso de uso "Consultar movimentações da conta bancária" que é utilizado por outros dois casos de uso: "Visualizar extratos" e "Emitir extrato".

Após a determinação dos casos de uso deve-se especificar seu comportamento. Isto é feito através de diagramas de sequências para a linha principal e outros para os casos variantes. Além disso deve ser especificado um diagrama de sequências para cada tipo de erro ou condição de exceção. Deve-se ter em mente que o tratamento dos erros é parte do caso de uso e deve

ser documentado juntamente com a linha principal (BOOCH; RUMBUGH; JACOBSON, 2006, p. 248).

Figura 7 – Diagrama de casos de uso modelado pelos requisitos de um sistema.



4.4 Documentação de Casos de Uso

A documentação de casos de uso é uma descrição textual que descreve uma unidade funcional do sistema. A Tabela 8 apresenta um exemplo simplificado de documentação de caso de uso para um sistema de controle de estoques. Deve-se notar que todos os casos de uso pertencentes ao diagrama de casos de uso devem possuir uma documentação equivalente. Isto se faz necessário porque somente os elementos gráficos não expressam os detalhes contidos no funcionamento.

Tabela 8 – Caso de uso: Manter Produtos (UC1).

Identificador	UC1	Nome	Manter Produtos
Ator principal	Gerente		
Interessados (*stakeholders*) e interesses			
• Proprietário: deseja que os produtos pagos pela empresa sejam vendidos e o investimento volte a ser capital com um acréscimo de lucro; • Gerente: deseja manter constantemente atualizados os preços e quantidades em estoque dos produtos; • Vendedor: deseja poder solicitar o registro da venda de um produto para que sua comissão seja calculada; • Caixa: deseja poder adicionar todos os produtos no momento da venda, sem nenhum tipo de transtorno; • Cliente: deseja comprar o produto			
Pré-condições	Estar autenticado no sistema		
Pós-condições	Produto cadastrado no SGBD		
Fluxo básico (cenário de sucesso)			
1. O usuário aciona o menu de cadastros de produtos; 2. O sistema apresenta a tela de cadastro de produtos; 3. O usuário preenche todos os dados do produto; 4. O usuário aciona a opção salvar; 5. O sistema verifica se o preço de venda é inferior ao preço de compra 6. Caso seja inferior, o sistema apresenta a mensagem "O preço de compra não pode ser maior que o preço de venda.", em seguida, volta ao passo 3 7. Caso seja superior, o sistema exibe a mensagem "Salvo com sucesso"			
Fluxos alternativos (extensões)			

1. O usuário aciona a opção listar
    a) O sistema apresenta a tela de listagem de produtos
    b) O usuário preenche, se necessário, o nome do produto que deseja encontrar
    c) O usuário aciona a opção Pesquisar
    d) O sistema busca e apresenta a lista dos produtos conforme o filtro da pesquisa
    e) O usuário seleciona um determinado produto
    f) O usuário aciona a opção Editar
    g) O sistema exibe a tela de cadastro do produto selecionado
    h) Volta ao Fluxo Principal no Passo 3
2. O usuário aciona a opção duplicar
    a) O sistema apresenta todos os dados do produto em um novo cadastro, sem o código
    b) Volta ao Fluxo Principal no Passo 3

Frequência de ocorrência	Aproximadamente 5 vezes por dia

Problemas em aberto

- Existe a necessidade de entendimento de impostos?
- Existem regras em relação as taxas?

4.5 Exercícios

1. Desenhe:

   a) um ator;

   b) um caso de uso;

2. O que é um caso de uso?

3. O que é especificado em um caso de uso?

4. A respeito dos casos de uso, julgue as alternativas abaixo como verdadeiras ou falsas:

   ( ) É necessário que haja implementação para criar um caso de uso.
   ( ) Fornecem uma compreensão comum do comportamento do sistema entre os desenvolvedores e as partes interessadas.
   ( ) Não podem ser utilizados para validação da arquitetura.
   ( ) Não são amplamente gerais e nem específicos.
   ( ) Se preocupa com o "como" devem ser feitas as coisas.

5. O que deve ser observado pelo projetista no momento da criação do sistema?

6. O que é assunto e qual a regra de nomenclatura de casos de uso?

7. Por que se diz que um ator pode ser um *hardware* ou um programa?

8. Os atores estão dentro ou fora do sistema? Justifique.

9. Qual o nome dado ao elemento visual que liga um ator a um caso de uso? Faça um desenho ligando um ator a um caso de uso.

10. O dono de uma empresa que vende refrigerantes deseja um *software* para planejar as melhores rotas de entrega dos pedidos. Para isso o representante comercial informa as vendas de produtos, ou seja, os lugares onde o caminhão de entrega deve passar. Como se trata de um problema complexo um computador de alta performance foi adquirido para o cálculo da melhor rota possível. Desta maneira o gerente de vendas solicita que o sistema calcule a melhor rota somente após o acumulo de uma boa quantidade pedidos. No momento da entrega o entregador informa através de um telefone que o

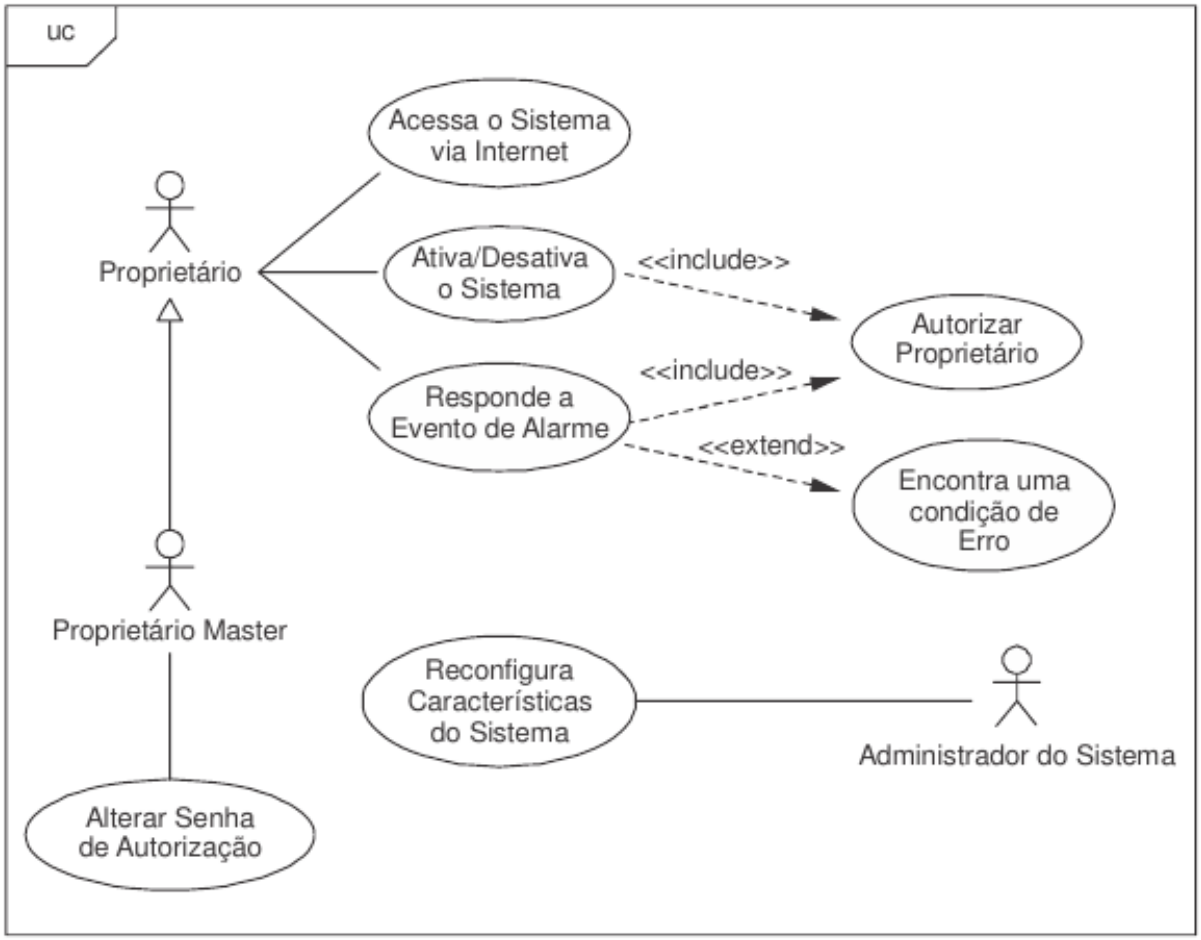
pedido foi entregue com sucesso. Faça um diagrama de caso de uso para representar o comportamento deste sistema.

11. Um comerciante deseja que um *software* seja desenvolvido. Neste domínio de problema existem três partes interessadas: o fornecedor, o cliente e o caixa. O fornecedor informa os preços atuais dos produtos que ele vende. O cliente visualiza seus débitos pela Internet. O caixa cadastra: os produtos, os pedidos feitos ao fornecedor, as sangrias (retiradas do caixa), os suprimentos (entradas no caixa), os pagamentos e as vendas. Todo o sistema deverá funcionar pela Internet, desta forma deve existir um mecanismo de autenticação e outro de recuperação de senha. Faça um diagrama de caso de uso para representar o comportamento deste sistema.

12. Um jornalista deseja um *website* para publicar notícias da região em que mora. Neste domínio de problema existem duas partes interessadas: o jornalista e o internauta. O internauta pode visualizar notícias, pesquisar notícias pelo título ou conteúdo, enviar um comentário em uma notícia, enviar a notícia para um colega via e-mail, enviar uma mensagem (página de contato) e visualizar todas as notícias de uma determinada categoria. O jornalista é quem redige as notícias no sistema, para isso ele deve se autenticar. Uma vez autenticado ele poderá: cadastrar uma categoria de notícia, cadastrar uma notícia, visualizar os comentários enviados pelos internautas, aprovar ou rejeitar os comentários, visualizar as mensagens enviadas pela página de contato, emitir a estatística de acesso do *website* e emitir a listagem de notícias mais visualizadas. Tudo que o internauta faz no sistema o jornalista também pode fazer. Faça um diagrama de caso de uso para representar o comportamento deste sistema.

4.6 Questões de Concurso

1. (TRT 2012) Considere a figura:



Na UML, este é um diagrama de

a) Classe.

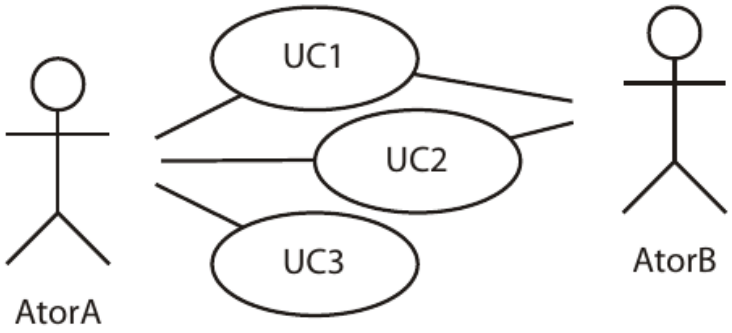
b) Sequência.

c) Caso de Uso.

d) Objetos.

e) Comunicação.

2. (SEFAZ-SC 2010) Considere a seguinte modelagem de casos de uso:



a) Ambos os elementos representados pelos atores AtorA e AtorB têm participação em todas as ocorrências do caso de uso UC1.

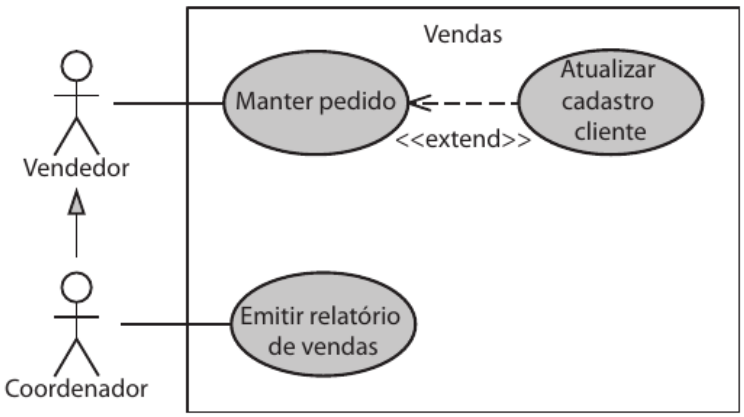
b) Apenas um dos elementos representados pelos atores (AtorA ou AtorB) tem participação em cada ocorrência do caso de uso UC2.

c) Ambos os elementos representados pelos atores AtorA e AtorB têm participação em todas as ocorrências do caso de uso UC3.

d) O elemento representado pelo ator AtorA pode ter participação em uma ocorrência do caso de uso UC1.

e) O elemento representado pelo ator AtorA tem participação em todas as ocorrências dos casos de uso UC1, UC2 e UC3.

3. (FATMA-SC 2012) Considere o diagrama de caso de uso da UML abaixo:



a) Decorrente do relacionamento de generalização entre os atores, o ator Coordenador executa uma tarefa comum com o ator Vendedor que é manter o pedido e atualizar o cadastro do cliente.

b) Decorrente do relacionamento de generalização entre os atores, o ator Vendedor executa uma tarefa comum com o ator Coordenador que é emitir relatório de vendas.

c) O relacionamento «extend» entre os casos de usos defne que o caso de uso Manter pedido pode, eventualmente, invocar o caso de uso Atualizar cadastro cliente.

d) O relacionamento entre o ator Coordenador com o caso de uso Emitir relatório de vendas é do tipo associação.

Assinale a alternativa que indica todas as afirmativas corretas.

a) É correta apenas a afirmativa 2.

b) São corretas apenas as afirmativas 1 e 4.

c) São corretas apenas as afirmativas 2 e 3.

d) São corretas apenas as afirmativas 3 e 4.

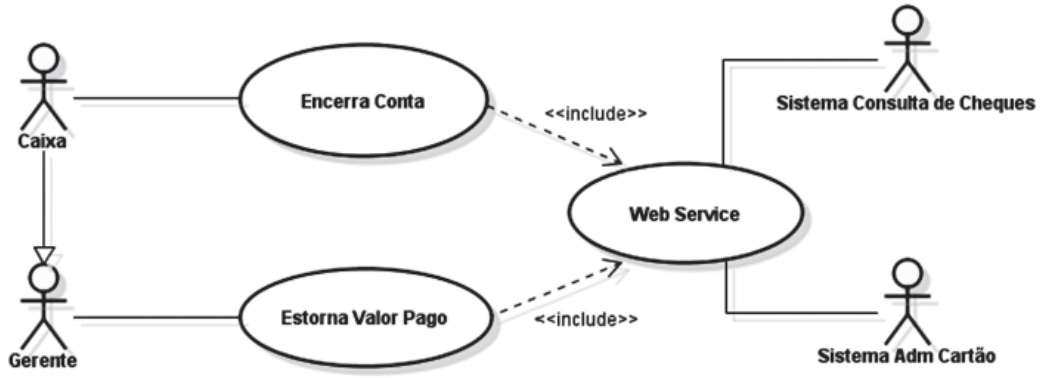
e) São corretas apenas as afirmativas 1, 3 e 4.

4. (CESGRANRIO 2012) Um restaurante contratou uma equipe para desenvolver um sistema de informação que auxilie nas tarefas diárias do negócio. Após um levantamento inicial, a equipe listou os seguintes requisitos:

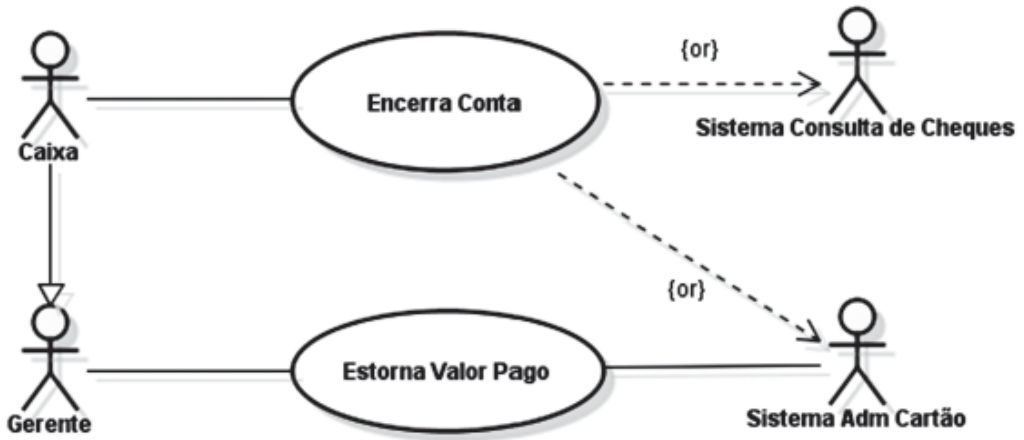
   - o caixa será responsável por encerrar uma conta e registrar o pagamento da mesma;
   - caso o pagamento seja feito com cheque, será necessário que o sistema do restaurante se comunique com o sistema de consulta de cheques do Serviço de Proteção ao Lojista para obter informações sobre o cliente;
   - caso o pagamento seja feito com cartão de crédito, será necessário que o sistema do restaurante se comunique com o sistema da administradora do cartão para obter autorização;
   - apenas o gerente terá acesso à função de estorno do valor pago. Caso a despesa tenha sido paga com cartão, será necessário se comunicar com o sistema da administradora;
   - tanto o sistema da administradora de cartões como o de consulta de cheques serão acessados via web service;

- o gerente também poderá encerrar uma conta.

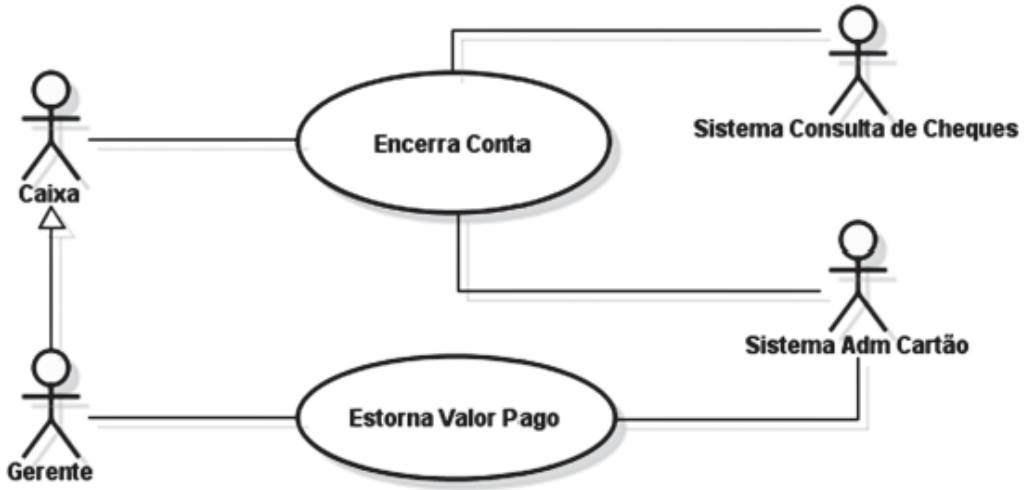
Qual diagrama de caso de uso descreve adequadamente os requisitos acima?



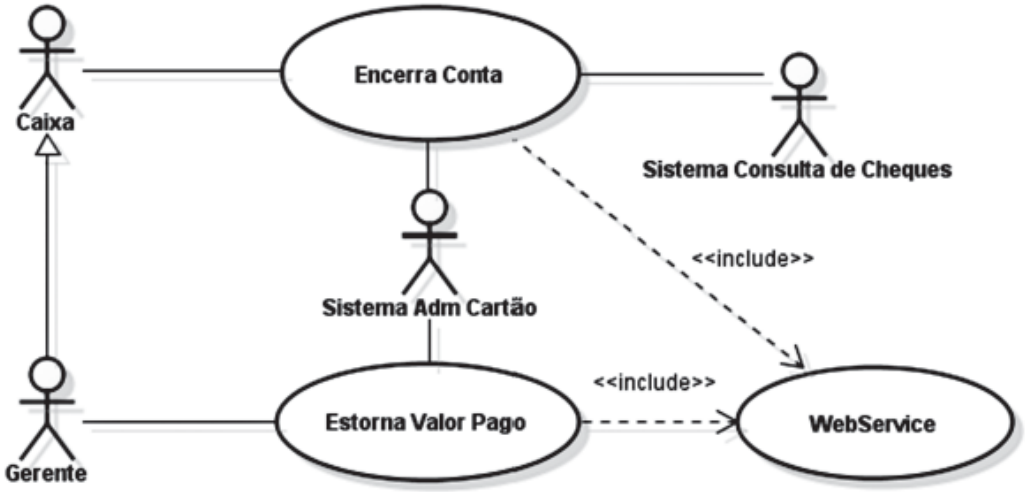
a)



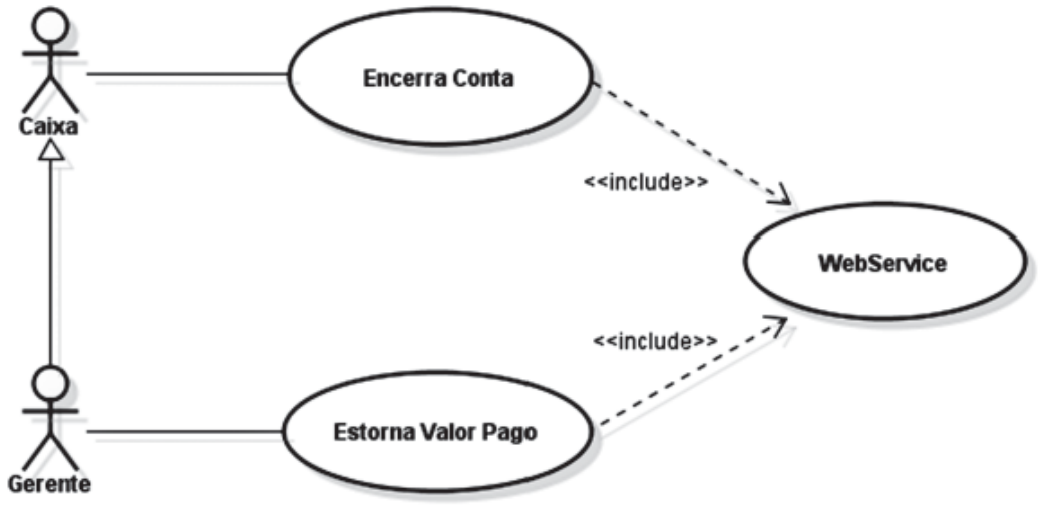
b)


Encerra Conta
Sistema Consulta de Cheques
Caixa
Sistema Adm Cartão
Estorna Valor Pago
Gerente

c)


Encerra Conta
Caixa
Sistema Consulta de Cheques
<<include>>
Sistema Adm Cartão
<<include>>
Estorna Valor Pago
WebService
Gerente

d)



e)

5. (CESPE 2012) Supondo que um sistema tenha sido desenvolvido e documentado de acordo com os conceitos da análise e do projeto orientado a objetos e tenha sido utilizada, como ferramenta para modelagem, a UML (Unified Modeling Language), versão 2.0, julgue os próximos itens.

   Considere um sistema de gerenciamento de documentos em que um diagrama da UML represente o caso de uso denominado "protocolar requerimento" e o caso de uso "protocolar retificação de requerimento". Nessa situação, a representação mais adequada é a que consiste em inserir um ponto de extensão no segundo caso de uso, a partir do qual ele será estendido pelo comportamento do primeiro.

   a) Certo

   b) Errado

6. (TRE-CE 2012) Na UML 2.0, representam comportamentos de um sistema, os diagramas de

   a) comunicação e de caso de uso.

   b) sequência e de implantação.

   c) componentes e de atividades.

   d) pacotes e de componentes.

e) atividades e de implantação.

7. (TRE-CE 2012) Em UML, os diagramas de Caso de Uso tem por objetivo

   a) representar os atributos e operações de uma classe ou objeto.

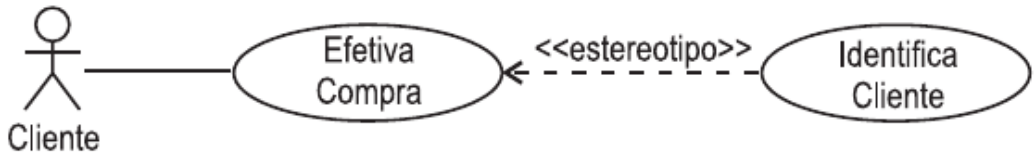
   b) mostrar o fluxo de mensagens de uma atividade do sistema para outra.

   c) capturar funcionalidades e requerimentos do sistema.

   d) exibir uma interação entre um conjunto de objetos e seus relacionamentos.

   e) representar o estado ou situação em que um objeto pode se encontrar no decorrer da execução de processos de um sistema.

8. (CESGRANRIO 2011) Durante o levantamento de um sistema, um analista registrou o seguinte requisito funcional: "A função de efetivação de uma compra deverá exigir que o cliente se identifique novamente para o sistema, caso o valor da transação ultrapasse o limite de crédito definido pela gerência." A partir desta declaração, o analista elaborou o diagrama de casos de uso UML 2.3 abaixo.



   Qual deve ser o estereótipo da relação entre os casos de uso Efetiva Compra e Identifica Cliente, de modo que esse diagrama expresse o requisito funcional descrito anteriormente?

   a) extend

   b) include

   c) inherits

   d) implements

   e) overrides

5 DIAGRAMA DE CLASSES

Booch, Rumbugh e Jacobson (2006, p. 107) afirmam que os diagramas de classes são frequentemente encontrados em sistemas orientados a objetos. Este diagrama apresenta os atributos e métodos das classes, as interfaces, as colaborações e seus relacionamentos. Trata-se de um diagrama estrutural que fornecem bases para outros diagramas como diagrama de componentes e diagrama de implantação.

5.1 Classes

As classes são estruturas que permitem armazenar dados e também executar comandos, através respectivamente dos recursos atributos e métodos. Devido a complexidade dos sistemas estas classes podem ser agrupadas em pacotes contendo classes sobre um mesmo assunto.

As classes são utilizadas para criar os objetos. Geralmente este conceito é difícil de entender no primeiro momento, por isso a Figura 8 apresenta uma analogia sobre a diferença de classes e objetos. As classes definem as características dos objetos, assim como a fôrma de trufas definem os formatos delas.

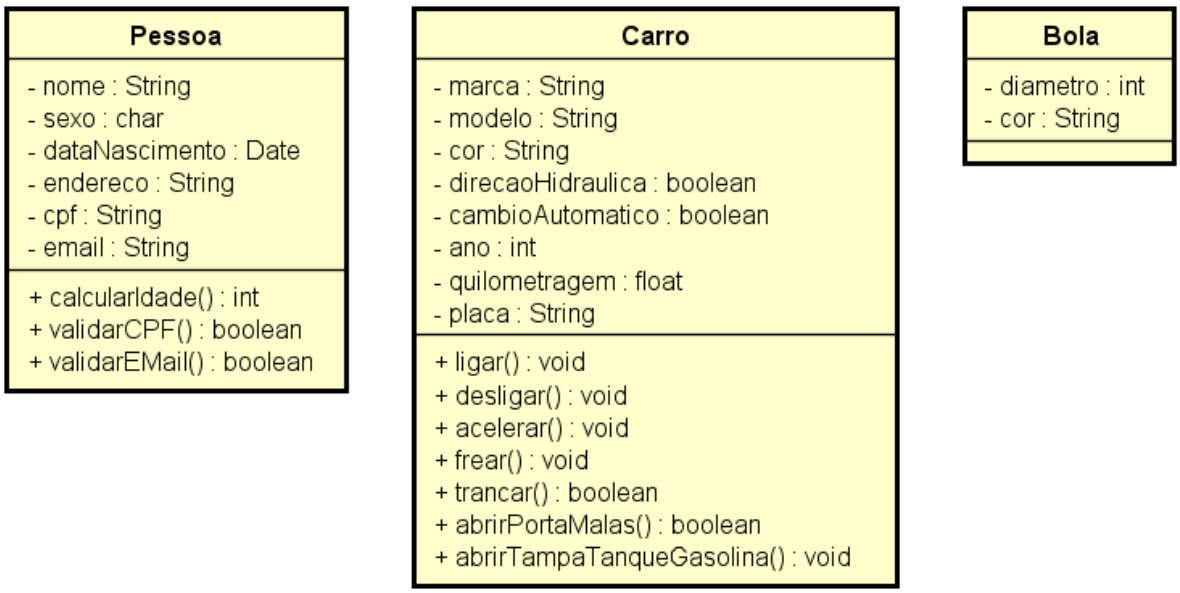
Figura 8 – Analogia de classes e objetos. A fôrma representa as classes enquanto que as trufas são os objetos criados através delas.

É importante perceber que mesmo sendo criadas da mesma fôrma, as trufas podem possuir diferentes características, como na imagem exitem trufas de chocolate preto e chocolate branco, e o enfeite da cobertura diferente, assim como o recheio interno. Estas diferenças são possíveis valores para os atributos do objeto.

A Figura 9 apresenta três classes usando a notação da UML. Nela cada classe é representada como um retângulo. Este possui três divisões, a primeira destinada ao nome da

classe, a segunda aos atributos e a terceira aos métodos. Uma classe pode não ter atributos e/ou métodos. Neste exemplo a classe *Bola* não possui métodos.

Figura 9 – Representação de classes na UML.



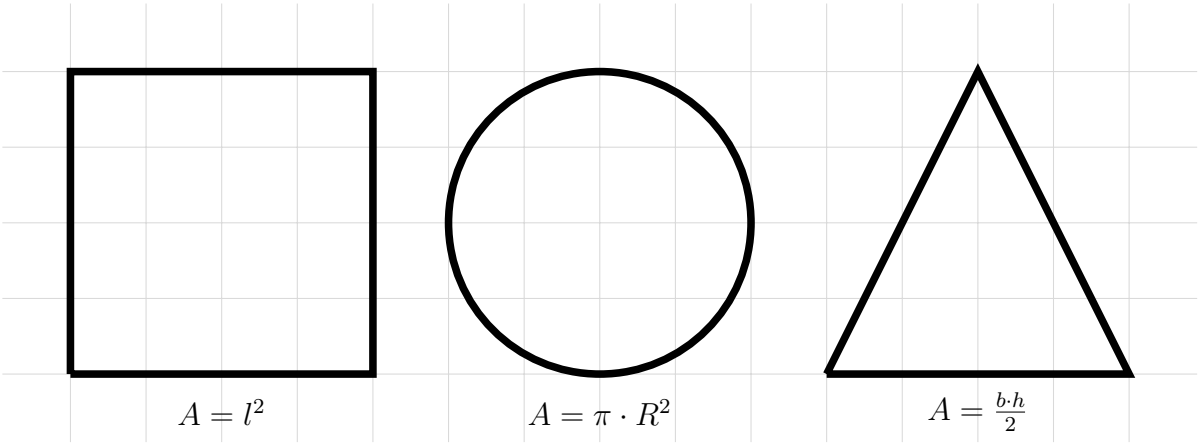
Um atributo é um tipo especial de variável que pertence a um objeto. No exemplo dado, a partir da classe *Pessoa* é possível criar várias pessoas, cada uma com um nome, sexo e data de nascimento diferentes. Assim, um atributo define o escopo das variáveis, e por isso podem ser chamados de variáveis de instância.

Os métodos por sua vez definem os comportamentos de um objeto. Nas linguagens de programação eles são classificados em funções e procedimentos. Assim, um método possui internamente a ele um algoritmo que pode modificar os valores das variáveis e resolver um determinado problema. Na classe *Pessoa* existe um método chamado *calcularIdade()*, este calcula a idade do indivíduo com base na data atual do sistema e o atributo data de nascimento, seu resultado é dado em anos, ou seja, um número inteiro.

5.2 Interfaces

Em algumas situações é necessário e mais organizado lidar com várias classes ao mesmo tempo. A Figura 10 apresenta um problema clássico que justifica o conceito de interfaces na orientação a objetos. Neste problema existem diferentes figuras geométricas, cada qual possui uma classe específica, mas é possível calcular a área de qualquer figura geométrica independentemente de qual seja.

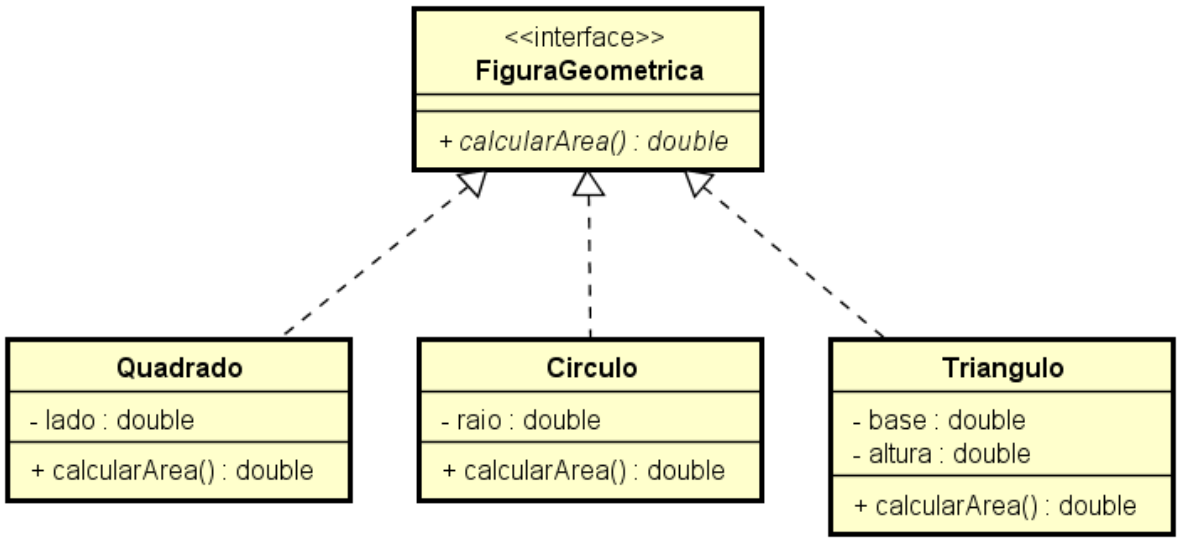
Figura 10 – Analogia sobre interfaces e sua importância.



A Figura 11 apresenta o projeto do problema usando interface. Neste definiu-se uma interface chamada *FiguraGeometrica* responsável por determinar que todas as classes que a implementam devem ter obrigatoriamente o método *calcularArea()*. O que determina que *FiguraGeometrica* é uma interface é o esteriótipo *«interface»* acima do nome dela.

Neste diagrama as classes *Quadrado*, *Circulo* e *Triangulo* realizam a interface *FiguraGeometrica*, ou seja, elas obedecem os padrões determinados por ela, tecnicamente implementam os métodos que ela define. Isto é simbolizado no diagrama com um relacionamento chamado realização, este é representado por uma linha tracejada com um triângulo não preenchido na extremidade.

Figura 11 – Exemplo de situação de uso de interface.



5.3 Relacionamentos

Existem três tipos de relacionamentos entre as classes: dependência, generalização e associação.

5.4 O diagrama de classes

O diagrama de classes pode ser utilizado para representar três propósitos: 1) modelar do vocabulário de um sistema; 2) modelar as colaborações simples; e 3) modelar o esquema lógico de um banco de dados (BOOCH; RUMBUGH; JACOBSON, 2006, p. 109 e 110).

5.4.1 Modelagem do vocabulário de um sistema

Consiste na utilização do diagrama para especificar quais abstrações fazem parte do sistema e quais não.

5.4.2 Modelagem das colaborações simples

Uma colaboração consiste no funcionamento em conjunto entre classes, interfaces e outros elementos para realizar um comportamento cooperativo, caracterizando-se por uma propriedade emergente das classes.

5.4.3 Modelagem do esquema lógico de um banco de dados

Pode-se criar um esquema que servirá de base conceitual de um banco de dados.

6 INTERAÇÕES

Em um sistema os objetos comunicam entre si para atingirem um determinado objetivo. As interações modelam os aspéctos dinâmicos das colaborações dentro de um sistema. Cada objeto executa um papel específico dentro de um sistema e um conjunto de objetos trabalham na execução de um determinado comportamento (BOOCH; RUMBUGH; JACOBSON, 2006, p. 211).

Os aspéctos dinâmicos são visualizados, especificados, construídos e documentados como fluxos de controle que podem abranger uma ou mais linhas de processamento, podendo envolver ramificações (condicionais), laços, recursão e concorrência. Existem duas abordagens para a modelagem de cada interação, a primeira com **ênfase no tempo** e a segunda com **ênfase na sequência das mensagens** (BOOCH; RUMBUGH; JACOBSON, 2006, p. 211).

As características de uma interação bem-estruturada são as mesmas de um algoritmo bem-estruturado: eficiêntes, simples, adaptáveis e compreensíveis (BOOCH; RUMBUGH; JACOBSON, 2006, p. 211).

Os *softwares* são criados com base em objetos os quais interagem entre si para manuesar corretamente os dados e transformá-los em informações úteis para as partes interessadas. Alguns sistemas são classificados como **reativos**, pois os objetos do sistema só interagem por meio de ações externas, este é o caso de um forno microondas, um arcondicionado e uma geladeira (BOOCH; RUMBUGH; JACOBSON, 2006, p. 212). No *software* os eventos externos podem ser: pressionar de um botão, movimento de um cursor, comando de voz entre outros.

A modelagem de elementos estáticos (**modelagem estrutural**) é realizada através do diagramas de classes e do diagrama de objetos. Através destes diagramas é possível visualizar, especificar, construir e documentar a estrutura dos componentes do sistema e seus relacionamentos (BOOCH; RUMBUGH; JACOBSON, 2006, p. 212).

A modelagem dos aspéctos dinâmicos é relizada através das interações. Uma interação define estaticamente um conjunto de estados para o comportamento. Todos os objetos envolvidos são adicionados na interação que fazem parte uma vez que são eles os agentes das ações. Na interação também são representadas as mensagens trocadas pelos objetos. Geralmente cada mensagem representa uma chamada a uma operação de um objeto ou envio de um sinal. Além disso as mensagens podem envolver a criação e destruição de objetos (BOOCH; RUMBUGH; JACOBSON, 2006, p. 212).

Uma interação pode ser utilizada para modelar o fluxo de controle de uma operação,

uma classe, um componente, um caso de uso ou um sistema. Através das duas abordagens de modelagem das interações, pode-se analisar a maneira como as mensagens são trocadas ao longo do tempo ou os relacionamentos estruturais entre os objetos, ou seja, como as mensagens são passadas no sistema (BOOCH; RUMBUGH; JACOBSON, 2006, p. 213).

6.1 Termos e Conceitos

Chama-se de **interação** um comportamento que contém um conjunto de mensagens trocadas entre os objetos para chegar a um determinado propósito. Uma **mensagem** é a especificação da comunicação entre dois objetos, ela contém a definição do resultado esperado e dos parâmetros necessários para o processamento dela (BOOCH; RUMBUGH; JACOBSON, 2006, p. 213).

Um conjunto de interações são acionadas para atingir um objetivo mais amplo, este conjunto é chamado de **contexto** (BOOCH; RUMBUGH; JACOBSON, 2006, p. 213 e 214).

Os objetos participantes de uma interação são elementos concretos. Em uma interação são encontradas instâncias de classes, componentes, nós e casos de uso. Além de instâncias indiretas de classes abstratas e interfaces. As interações definem uma sequência dinâmica nas mensagens passadas entre os objetos (BOOCH; RUMBUGH; JACOBSON, 2006, p. 214).

6.1.1 Vínculos e Conectores

Um **vínculo** é uma conexão semântica entre objetos. Ele determina o caminho pelo qual um objeto envia uma mensagem para outro ou para ele mesmo (BOOCH; RUMBUGH; JACOBSON, 2006, p. 215).

Chama-se de **papel** um objeto protótipico. Um vínculo protótipico chama-se de **conector** (BOOCH; RUMBUGH; JACOBSON, 2006, p. 216).

6.1.2 Mensagens

Uma vez elencados os objetos e vínculos que participam de uma interação tem-se um modelo estático que poderia ser representado por um diagrama de objetos. No entanto quando se deseja analisar as mudanças de estados destes objetos ao longo do tempo, haverão mensagens entre este objetos para que tais mudanças ocorram. Desta forma tem-se que uma **mensagem** é a especificação de uma comunicação entre objetos (BOOCH; RUMBUGH; JACOBSON, 2006, p. 217).

REFERÊNCIAS

BEZERRA, E. **Princípios de Análise e Projeto de sistemas com UML**. 3. ed. Rio de Janeiro: Elsevier, 2014. ISBN 8535226265.

BOOCH, G.; RUMBUGH, J.; JACOBSON, I. **UML: Guia do Usuário**. 2. ed. Rio de Janeiro: Campus, 2006.

MACORATTI, J. C. **Conceitos : Especificação de requisitos**. 2012. Disponível em: ⟨http://www.macoratti.net/07/12/net_fer.htm⟩. Acesso em: 02 mar. 2012.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 1995. ISBN 85-346-0237-9.

ROCHA, C. C. **Joint Application Design (JAD)**. 2017. Disponível em: ⟨http://www.matera.com/br/2014/02/11/joint-application-design-jad/⟩. Acesso em: 10 fev. 2017.

SOMMERVILLE, I. **Engenharia de software**. 8. ed. São Paulo: Pearson - Addison Wesley, 2007. ISBN 978-85-8863-928-7.