UNIVERSIDADE PITÁGORAS UNOPAR ANHANGUERA

POLO IBIRITÉ (CENTRO) / MG

SEMESTRE 2024/2

Relatório de Aula Prática:

Diagrama de Caso de Uso para a Modelagem de um Sistema Bancário

ALUNO: DENIS DA MATA OLIVEIRA

RA: 3821926601

CURSO: ENGENHARIA DE SOFTWARE - BACHARELADO

MODALIDADE: 100% ONLINE

DISCIPLINA: ANÁLISE E MODELAGEM DE SISTEMAS

PROFESSOR: VANESSA MATIAS LEITE

DATA: 07/11/2024

INTRODUÇÃO

A linguagem são regras e símbolos usados pra transmitir informação, e é tão importante, que é fundamental para o desenvolvimento intelectual e cognitivo do ser humano [1]. A linguagem que nós conhecemos, como português e o inglês, é chamada de linguagem natural, que é mais complexa e menos rigorosa, enquanto a matemática, que tem uma abrangência menor, é mais rigorosa e é chamada de linguagem formal. Usando o português, nós temos que "um mais um é igual a dois", já com a matemática nós temos que "1+1=2". Com a matemática é muito mais fácil de lidar com quantidades que são mensuráveis.

Existem vários tipos de linguagens, mas aqui estamos interessados em um caso particular de linguagem, usada na análise e desenvolvimento de software, chamada de linguagem visual, cuja a representação é baseada em imagens e diagramas. A **Figura 1** mostra as várias linguagens existentes na engenharia de software, e mostra onde a linguagem visual se encaixa.

Figura 1: *Linguagens do desenvolvimento de software. A informação transmitida com linguagem de alto nível é mais acessível, mas é menos detalhada. O tradutor é a pessoa responsável por intercalar a informação entre o baixo e o alto nível.* *Imagem tirada da internet.*

Como as linguagens visuais se baseiam em imagens, elas carregam muita informação e são fáceis de serem processadas pelo cérebro humano. É como diz o ditado popular "uma imagem vale mais do que mil palavras". Além disso, elas podem ser usadas em qualquer área, não só na computação ou programação. Por outro lado, ela é menos rigorosa, e é representada com "desenhos", mais trabalhoso e difícil de criar e ser transmitido por teclado. Toda linguagem possui regras a serem seguidas, e essas regras derivam das demandas naturais das pessoas envolvidas, que se refletem na estrutura da linguagem.

Dentro das linguagens visuais, temos aquela que é representada com diagramas e focada em atores e funcionalidades, a linguagem de caso de uso, também chamada de diagrama de caso de uso. O diagrama de caso de uso é o mais básico e usado em análise e desenvolvimento de sistema. Os diagramas, de modo geral, estão presentes em todo o desenvolvimento de software, mas o diagrama de caso de uso é mais presente na fase inicial do projeto e no levantamento de requisitos, pois ele é mais geral e foca nas pessoas e funcionalidades, sem se preocupar com os detalhes de implementação. O diagrama de caso de uso é tão básico, que no gerenciamento de processo do tipo Processo Unificado, PU ou RUP, o diagrama de caso de uso é considerado o diagrama principal.

As pessoas são as partes mais importantes em qualquer projeto, e o diagrama de caso de uso foca nelas. As pessoas podem ser clientes, gerentes de software, bots ou terceiros, que no diagrama de caso de uso são representados por um boneco palito. Os diagramas se dividem em comportamentais (temporais/dinâmicos) ou estruturais (estáticos), e o diagrama de caso de uso é considerado um diagrama estrutural, pois não se preocupa com a ordem dos acontecimentos, ou como eles acontecem, e sim quais são as relações existentes, entre os atores e as funcionalidades. O diagrama de caso de uso é bem simples, é basicamente boneco palito, balão e seta, mas cada balão, que representa uma funcionalidade, tem toda uma complexidade por trás. Um balão do diagrama de caso de uso pode dar origem a outro diagrama, por exemplo, um diagrama de atividade.

As relações do diagrama de caso de uso podem ser representadas por setas preenchidas ou tracejadas. As setas preenchidas são as relações simples, e as tracejadas são acompanhadas de legenda e se dividem em dois tipos. Uma relação tracejada do tipo `include` significa que uma funcionalidade tem que ser executada, caso outra funcionalidade seja executada, enquanto a relação tracejada `extend` é opcional. Um exemplo de relação do tipo `include` é a quitação de dívida, ao fechar uma conta bancária, pois para fechar a conta bancária, necessariamente, o cliente tem que estar quite com o banco. Já na relação `extend`, um exemplo seria a tradução de uma página da internet pelo navegador, que é realizada a gosto do usuário.

Podemos construir uma casa sem uma planta? A resposta é, sim, você pode, mas é bem menos seguro. Nisso que entra a engenharia e planejamento através da modelagem. A *Unified Modeling Language* (Linguagem de Modelagem Unificada), ou UML, é uma linguagem de modelagem visual usada para construir e documentar, de forma visual e padronizada, um sis-

tema de software. Ela é baseada em diagramas e notações para representar diferentes aspectos do sistema, como estruturas e comportamentos. Além da UML poder ser usada para construir diagramas de caso de uso, ela pode ser usada para uma grande variedade de outros tipos de diagramas. Existe vários sites online gratuitos que servem como ferramenta para construir diagramas de caso de uso. O PlantUML, por exemplo, permite criar vários tipos de diagramas usando scripts, que combinado com inteligência artificial, se torna uma ferramenta muito poderosa, como exemplificado no **Apêndice A**. Outra ferramenta de diagramação textual muito usada é o Mermaid, que é reconhecido por padrão em arquivos markdown (.MD) e vários outros sites. Aqui seguiremos a sugestão do roteiro da aula prática e usaremos o Visual Paradigm Online, que tem uma versão gratuita com muitos recursos interativos.

OBJETIVO

- Desenvolver um diagrama de casos de uso utilizando os conhecimentos e práticas da UML.

MÉTODO E RESULTADOS

Vejamos o que foi pedido no roteiro:

> **Problema Proposto:**
> Desenvolva um diagrama de casos de uso para um sistema de bancário, levando em consideração os seguintes requisitos:
> • O cliente pode abrir e encerrar contas, para isso, ele deverá procurar um funcionário no banco.
> • O cliente pode abrir uma conta do tipo especial ou poupança.
> • O cliente pode depositar ou sacar dinheiro, estas funcionalidades podem ser feitas no caixa eletrônico.
> • O cliente pode emitir o saldo ou extrato da sua conta, estas funcionalidades podem ser feitas no caixa eletrônico.
> • Para o cliente encerrar a sua conta, seu saldo deve estar zerado.
> • Cada movimentação realizada deve ser registrada.

Na **Figura 2** é mostrado o resultado obtido usando o Visual Paradigm Online [7]. Com

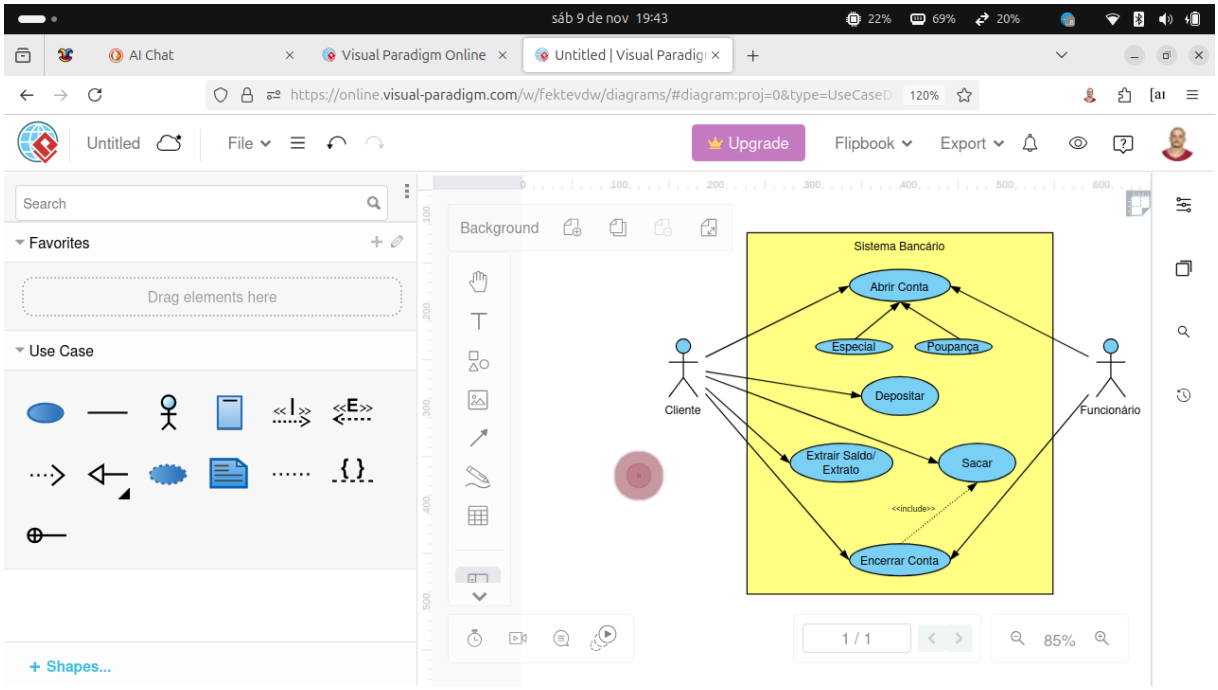

Figura 2: *Resultado da resolução do problema da modelagem de um sistema bancário usando diagrama de caso de uso [7].*

a imagem é bem mais rápido e fácil de visualizar as pessoas envolvidas, funcionalidades principais e a relação entre cada um. Esse diagrama é tão geral que pode ser usado como uma referência para a modelagem de processos de negócio. O nosso diagrama tem apenas duas pessoas, mas poderia ter várias pessoas, por exemplo, gerente de software, serviço de terceiros que poderia envolver uma burocracia, gerente do banco se uma funcionalidade for muito eventual, e assim por diante. Como pode ser visto na **Figura 2**, na aba lateral esquerda identificada com "*Use Case*", a quantidade de elementos básicos usados no diagrama de caso de uso é bem pequena, apenas temos o boneco palito, o balão das funcionalidades, conectivos e mais alguns elementos para detalhar mais o diagrama. Em outros diagramas, como diagrama de classe, a quantidade de elementos é muito maior. O Visual Paradigm Online tem várias nuances, por exemplo, na **Figura 2** foi usado a seta abaixo do "*Background*", abaixo do ícone de imagem, ao invés de usar a seta da aba lateral esquerda. Esta seta sempre é reta e deixa o diagrama mais bonito, que é viável no nosso caso. Apesar das nuances, o site é muito interativo e basta brincar um pouco com ele para ganhar familiaridade.

Para o cliente sacar o dinheiro, ele deve ir até o banco, a pé ou de carro, então por que não criamos um balão para essa atividade (fora do sistema bancário)? Por que não coloca-

mos a função de depositar e sacar no mesmo balão? Aqui estamos interessados num sistema bancário, que é um software. O diagrama de caso de uso tem, como uma de suas referências, a programação orientadas a objetos, que é o paradigma mais popular, e diferente de um diagrama de classe, por exemplo, o diagrama de caso de uso nos fornece uma ideia do processo de negócio, mas com foco os atores, serviços do sistema e como os atores são atendidos por esses serviços. Então, a estrutura do diagrama segue essa demanda. Os balões lembram muito um diagrama de classe, sem os detalhes de implementação. A conta "Especial" ou "Poupança" seriam subclasses de "Abrir Conta". A relação dentre "Encerrar Conta" e "Sacar" dinheiro é semelhante a relação de dependência do diagrama de classe, que é representada, também, por uma seta pontilhada. Separamos o "Sacar do "Depositar" por causa de "Encerrar Conta", mas observe que essas duas funcionalidades poderiam ser abstraídas em uma só funcionalidade chamada "Transação" ou "Operação Bancária". Contudo, esta funcionalidade, além de não ter sido solicitada pelo roteiro da aula prática, não teria relação direta com nenhum ator, se tornado menos relevante, ainda, para o nosso diagrama.

CONCLUSÃO

Com essa aula prática pudemos nos familiarizar com o diagrama de caso de uso, que é muito interessante, pois é acessível, ágil e pode ser usado em várias áreas, não só para desenvolvimento de software. O Visual Paradigm é bastante fácil de usar e foi bem sucedido a construção manual do diagrama com ele, contudo, também podemos usar scripts gerados por inteligência artificial para construir diagramas bem maiores e complexos, como foi mostrado no **Apêndice A**. Os diagramas podem ser usados para comunicação informal e com uma variedade de interpretações, mas a lógica seguida aqui foi para modelar um software de sistema bancário com foco nos atores, funcionalidades que pertencem ao sistema e as relações entre cada um. Pudemos notar similaridades entre o diagrama de caso de uso com a modelagem de processo de negócio e programação orientada a objeto. O trabalho foi fácil e as aulas deram uma boa base para a aula prática, por isso aproveitamos para nos aprofundar na ciência da linguagens e vários conteúdos relacionados, sem perder a coerência.

REFERÊNCIAS

Como pode ser visto pelo estilo de escrita e a profundidade (com uma citação a Vygotsky),

o texto do relatório não foi gerado pela inteligência artificial ChatGPT, mas o ChatGPT [4] foi usado em diversos momentos para tirar e dúvidas e esclarecer conceitos. Somente o código do **Apêndice A** que foi escrito e gerado exclusivamente pelo ChatGPT.

[1] VYGOTSKY, L. S. (1978). **A Formação Social da Mente: O Desenvolvimento dos Processos Psicológicos Superiores**. Trad. de L. M. de A. F. de Almeida. São Paulo: Martins Fontes.

[2] Canal do YouTube "Bóson Treinamentos", **Curso de UML - O que são Casos de Uso**.

[3] Canal do YouTube "Bóson Treinamentos", **Curso de UML - O que são Diagramas de Casos de Uso**.

[4] Inteligência artificial ChatGPT, **https://chatgpt.com/**.

[5] Site da internet "plantuml.com", **PlantUML Web Server**.

[6] Site da internet "Visual Paradigm Online", **https://online.visual-paradigm.com/**.

[7] Site da internet "Visual Paradigm Online", resultado público para consulta, **https://online.visual-paradigm.com/community/share/untitled-1ygp6xemcz**.

APÊNDICE A

Exemplo simples de script fornecido pelo ChatGPT para construir um diagrama de caso de uso com o PlantUML [5]:

```
@startuml
left to right direction

!define USECASE(x) usecase x as "x"

actor Jogador
actor Administrador
actor Convidado

rectangle "Aplicativo de Jogo" {
        USECASE(Iniciar_Jogo)
        USECASE(Jogar)
        USECASE(Pausar_Jogo)
        USECASE(Ver_Placar)
        USECASE(Criar_Conta)
        USECASE(Login)
        USECASE(Convidar_Amigo)
        USECASE(Receber_Premio)
        USECASE(Ver_Instrucoes)
}

Jogador --> Iniciar_Jogo
Jogador --> Jogar
Jogador --> Pausar_Jogo
Jogador --> Ver_Placar
Jogador --> Criar_Conta
Jogador --> Login
```

```
Jogador --> Convidar_Amigo
Jogador --> Ver_Instrucoes

Administrador --> Ver_Placar
Administrador --> Receber_Premio

Convidado --> Criar_Conta
Convidado --> Login

' Relacionamentos include
Iniciar_Jogo .down.> Login : <<include>>
Jogar .down.> Ver_Instrucoes : <<include>>

' Relacionamentos extend
Jogar .up.> Pausar_Jogo : <<extend>>
Convidar_Amigo .up.> Receber_Premio : <<extend>>
@enduml
```