Consumo Médio - Modelagem

Giulia Ferraro[1], Isabella Pagnoncelli[1]
[1]Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Avenida Ipiranga, 6681 Partenon - Porto Alegre - RS - Brasil

1. Introdução

Este relatório tem como objetivo descrever o processo de modelagem do problema URI 1014 - Consumo retirado do site *URI Online Judge* [URI Online Judge]. Neste documento iremos citar as especificações do problema a ser modelado bem como mostrar os modelos de duas formas diferentes: modelo inclusivo e modelo independente de plataforma, apontando a descrição de cada formato e seus benefícios. Os arquivos de cada etapa deste relatório estão contidos no repositório do *GitHub* (https://github.com/isabellapagnon/Trabalho1_Modelos).

2. Enunciado

O problema a ser modelado neste relatório é o de número 1014 da plataforma *URI Online Judge* [URI Online Judge]. Segundo a plataforma, o programa deverá realizar o cálculo do consumo médio de um veículo, ou seja, quantos quilômetros ele consegue rodar com apenas um litro de combustível.

Neste programa, a entrada será um valor inteiro representando a distância total percorrida e um valor real representando o total de combustível gasto. Após os dados de entrada serem registrados, a distância total percorrida será dividida pelo total de combustível gasto e a saída do programa será quantos quilômetros o veículo consegue rodar com apenas um litro de combustível.

3. Modelo inclusivo

O primeiro modelo a ser apresentado será o modelo inclusivo que foi representado utilizando o diagrama de casos de uso abaixo.

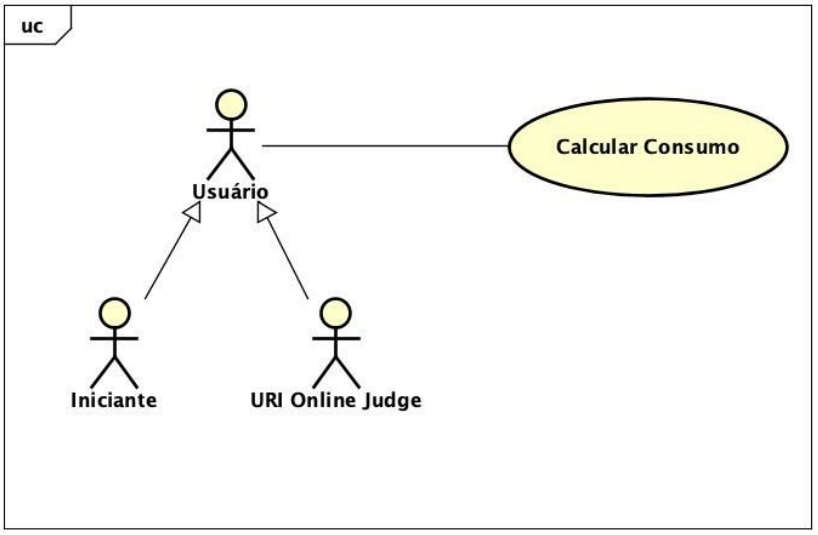

Figura 1: Diagrama de casos de uso do problema de consumo médio.

O modelo inclusivo é chamado assim pois ele tem a intenção de incluir os *stakeholders* na hora do planejamento do desenvolvimento do *software* [Agile Modeling]. A maioria dos *stakeholders* de algum projeto não tem muito conhecimento de termos de computação como classes e atributos então para ficar compreensível o desenvolvimento de um projeto para essas pessoas que são tão importantes é usado o diagrama de casos de uso.

A partir da interpretação do diagrama fica claro que o programa tem dois usuários, o Iniciante que é quem enviará uma tentativa de resolução do problema para a plataforma e o *URI Online Judge* que realizará a execução automatizada do código do Iniciante e avaliará se está de acordo com o que deveria ser.

Base Sequence	Usuário informa valor de X. Usuário informa valor de Y. Sistema calcula e apresenta o valor de consumo = X/Y.

Figura 2: Descrição do diagrama de casos de uso.

Além do diagrama de casos de uso é criado uma descrição para cada caso para explicar melhor o funcionamento dele como pode ser visto na Figura 2. O programa contém apenas um caso de uso que é Calcular Consumo e com a descrição para ajudar é possível entender que o usuário informará o valor de *X* (distância percorrida) e o valor de *Y* (total de combustível) e o sistema irá calcular o valor do consumo médio do veículo dividindo o valor de *X* pelo valor de *Y* e irá retornar o consumo para o usuário.

4. Modelo independente de plataforma

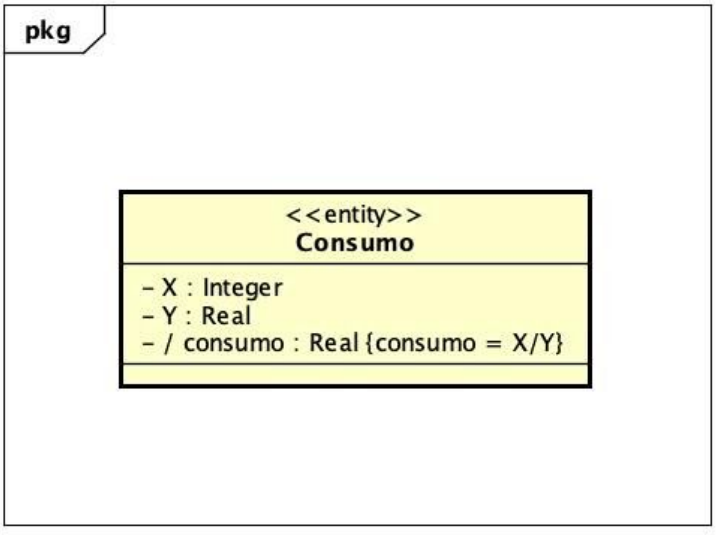

Figura 3: Diagrama de classes do problema de consumo médio.

Acima está apresentado o modelo independente de plataforma para o problema do consumo médio de um veículo. Um modelo independente de plataforma, segundo Oliveira (2011, p.10),

> "É um modelo independente das tecnologias da solução, ele apresenta uma perspectiva do modelo de negócio. Expressa as regras de negócio e as funcionalidades do *software* já com alguns aspectos computacionais, como por exemplo, questões sobre persistência, transação e segurança. Entretanto, os detalhes específicos sobre a implementação desses serviços não estão presentes."

Para representar um modelo independente de plataforma corretamente, foi utilizado a plataforma *Astah Professional* para desenvolver um diagrama de classes com seus atributos sendo de tipos primitivos. O atributo *X* representa a distância total percorrida e o atributo *Y* representa o gasto total de combustível e eles estão representados, respectivamente, pelos tipos primitivos *Integer* e *Real* pois assim não seguem o nome específico de nenhuma linguagem ou plataforma podendo ser facilmente compreensível por qualquer pessoa que estiver interpretando o diagrama.

O atributo consumo possui um valor inferido nele, no caso, a divisão da distância total percorrida pelo total de combustível gasto e para não usar um vocabulário específico de alguma linguagem de programação como *função* ou *método*, o cálculo necessário para obter o valor do consumo médio do veículo foi colocado entre *chaves*.

5. Lista de verificação

Para garantir que o relatório esteja completo e sem erros foi criado uma lista de verificação que pode ser verificada abaixo:

Pergunta	Verificação
O diagrama utiliza apenas tipos primitivos independentes da plataforma?	**Sim**
O artigo não possui erros gramaticais?	**Sim**
O artigo está formatado de acordo com as normas?	**Sim**
O repositório do GitHub foi criado?	**Sim**
Os aquivos .astah foram colocados corretamente na pasta destinada do GitHub?	**Sim**
Os modelos foram criados seguindo as especificações gerais?	**Sim**
A versão final do artigo foi colocada no GitHub?	**Sim**
O modelo foi criado utilizando a ferramenta correta (astah)?	**Sim**
O modelo possui os parâmetros corretos de entrada e de saída?	**Sim**

Figura 4: Lista de verificação do trabalho.

Os pontos que foram abordados na lista são abrangentes pois era preciso verificar não apenas se o diagrama está correto mas se o relatório também está sem erros e se o repositório no *GitHub* está completo. Por isso, na lista de verificação é averiguado tanto erros gramaticais quanto se os arquivos foram criado na plataforma correta e colocados no *GitHub* assim como se os modelos estão seguindo as especificações corretas.

6. Conclusão

A criação de modelos quando estamos programando nos traz inúmeros benefícios, a modelagem acima é para um problema pequeno e é possível compreender e implementar ele sem criar modelos mas quanto mais os problemas expandem, mais os modelos podem nos ajudar a ter uma visão melhor do que estamos fazendo e podemos implementar o problema com menos chances de erros.

A partir dos modelos criados é possível ter uma previsão de quantas horas será preciso para implementar o programa, bem como qual vai ser o recurso financeiro necessário para isso. Além disso, podemos ter uma nova visão do problema e entender ele melhor descobrindo maneiras mais eficazes de implementar ele pois os modelos deixam mais claro vários detalhes do programa que sem os modelos seriam muito abstratos. Nem sempre é possível fazer os modelos antes da implementação do programa mas está claro que os modelos só tem a acrescentar quando estamos desenvolvendo e devemos nos tornar mais familiarizados deles.

Referências

URI Online Judge. 1014 - Consumo. `https://www.urionlinejudge.com.br/judge/pt/problems/view/1014`. [Online; accessed 30-August-2019].

Agile Modeling. Inclusive Modeling. `http://agilemodeling.com/essays/inclusiveModels.htm`.[Online; accessed 30-August-2019].

Oliveira, Thiago Araújo Silva de. (2011). Geração de código estrutural implantável em nuvens a partir de modelos de componentes independentes de plataforma. *Universidade Federal de Pernambuco*, page 10.