Trabalho 1 - Engenharia de Software Orientada a Modelos

Pedro Chem e Rafael Almeida de Bem

30 de agosto de 2020

Sumário

Introdução

O presente relatório tem como objetivo documentar o processo de modelagem de dados para um enunciado de programação. Este projeto foi requisitado pelo Profesor Marco Aurélio Souza Mangan para a disciplina de Engenharia de Software Orientada a Modelos no curso de Ciência da Computação, Escola Politécnica, PUC-RS.

O trabalho envolve dois modelos. O primeiro é um modelo inclusivo, mais especificamente um diagrama de casos de uso 1.1 . O segundo é um modelo independente de plataforma, (respectivamente, inclusive model e platform independent model - PIM) 1.3, juntamente dos tipos primitivos de dados 1.2.

O enunciado de um exercício de programação escolhido foi: "Write an algorithm that reads two floating values (x and y), which should represent the coordinates of a point in a plane. Next, determine which quadrant the point belongs, or if you are over one of the Cartesian axes or the origin (x = y = 0)."Retirado do site URI Judge [3].

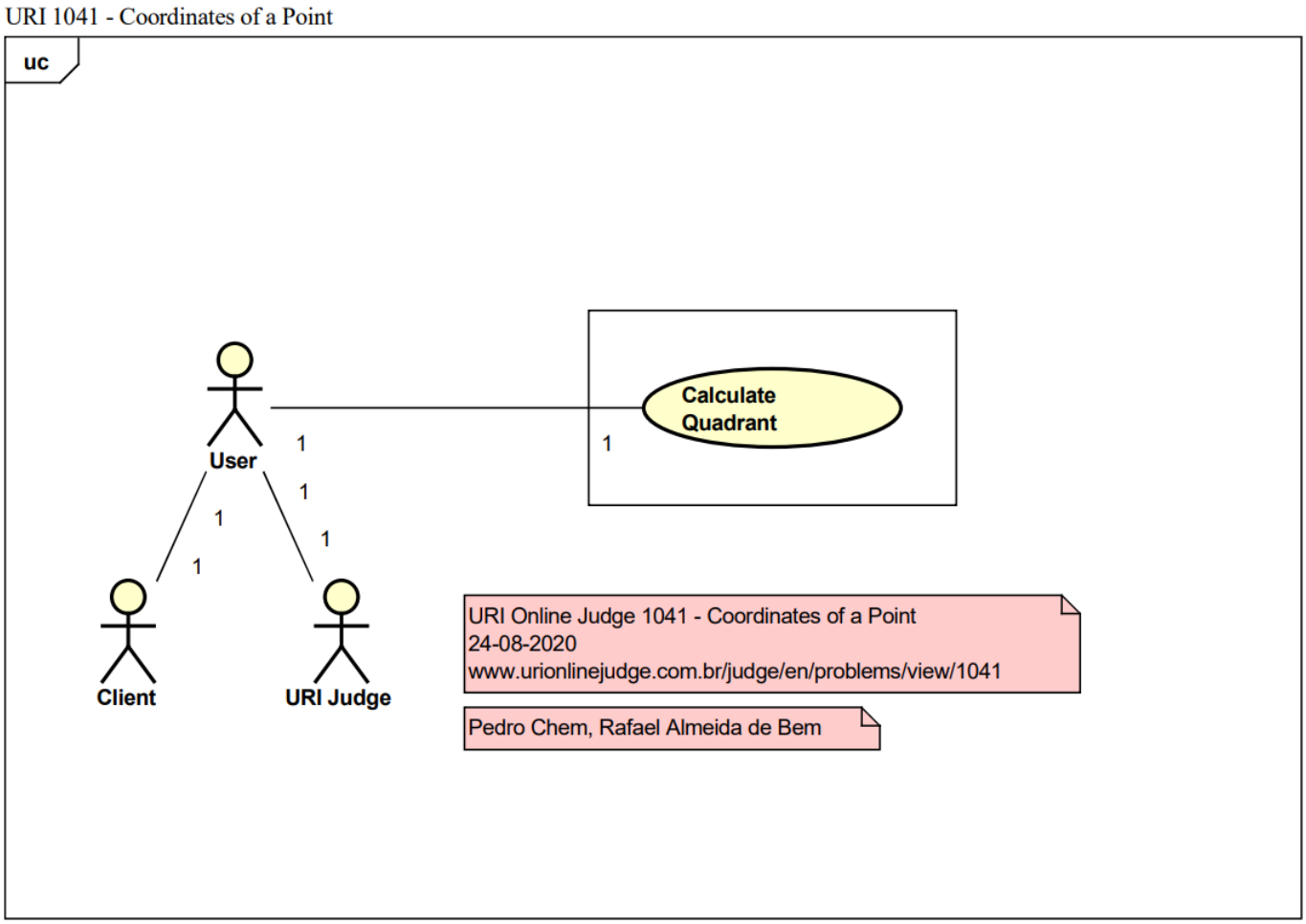

Figura 1.1: Diagrama Casos de Uso

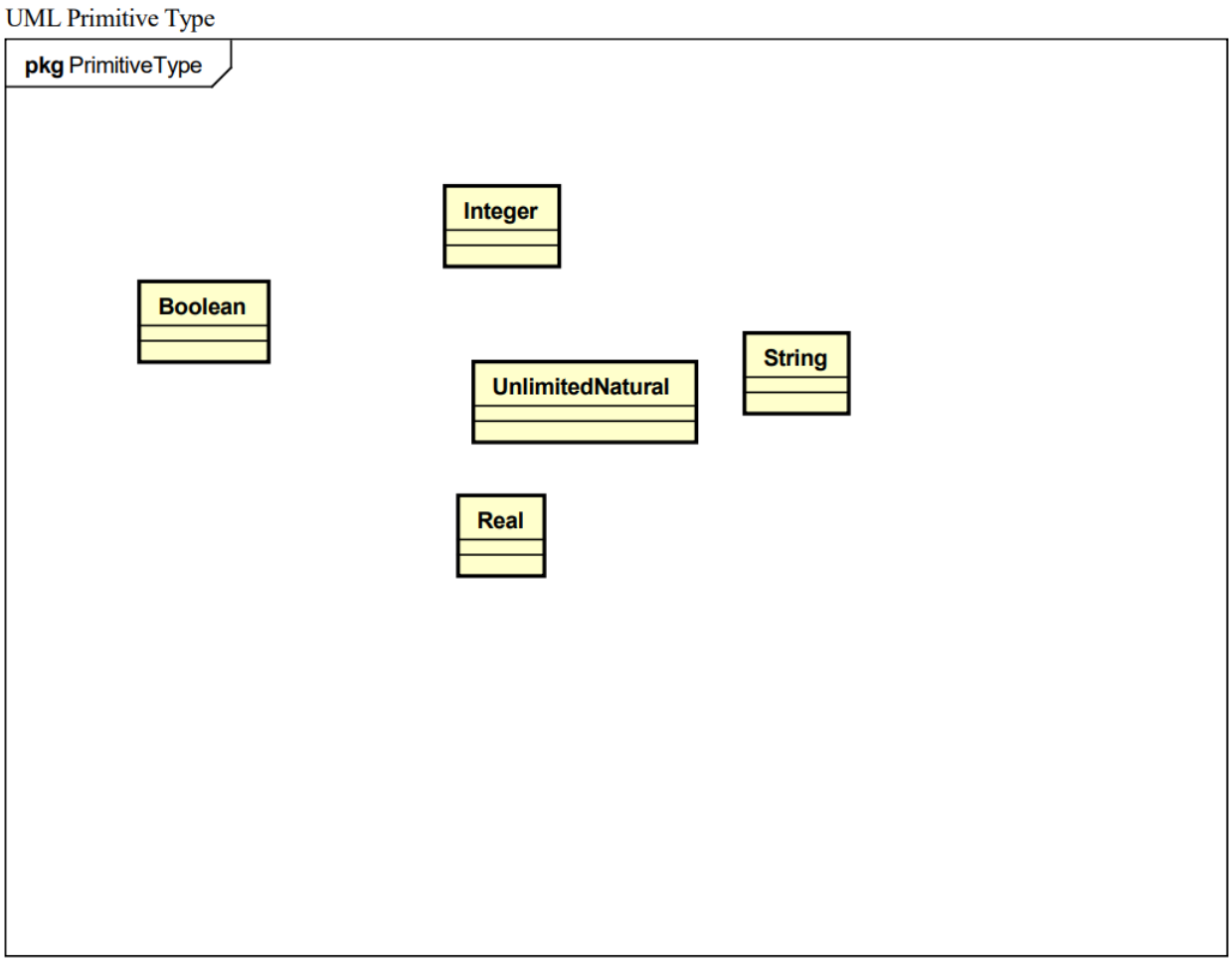

Figura 1.2: Tipos Primitivos de Dados

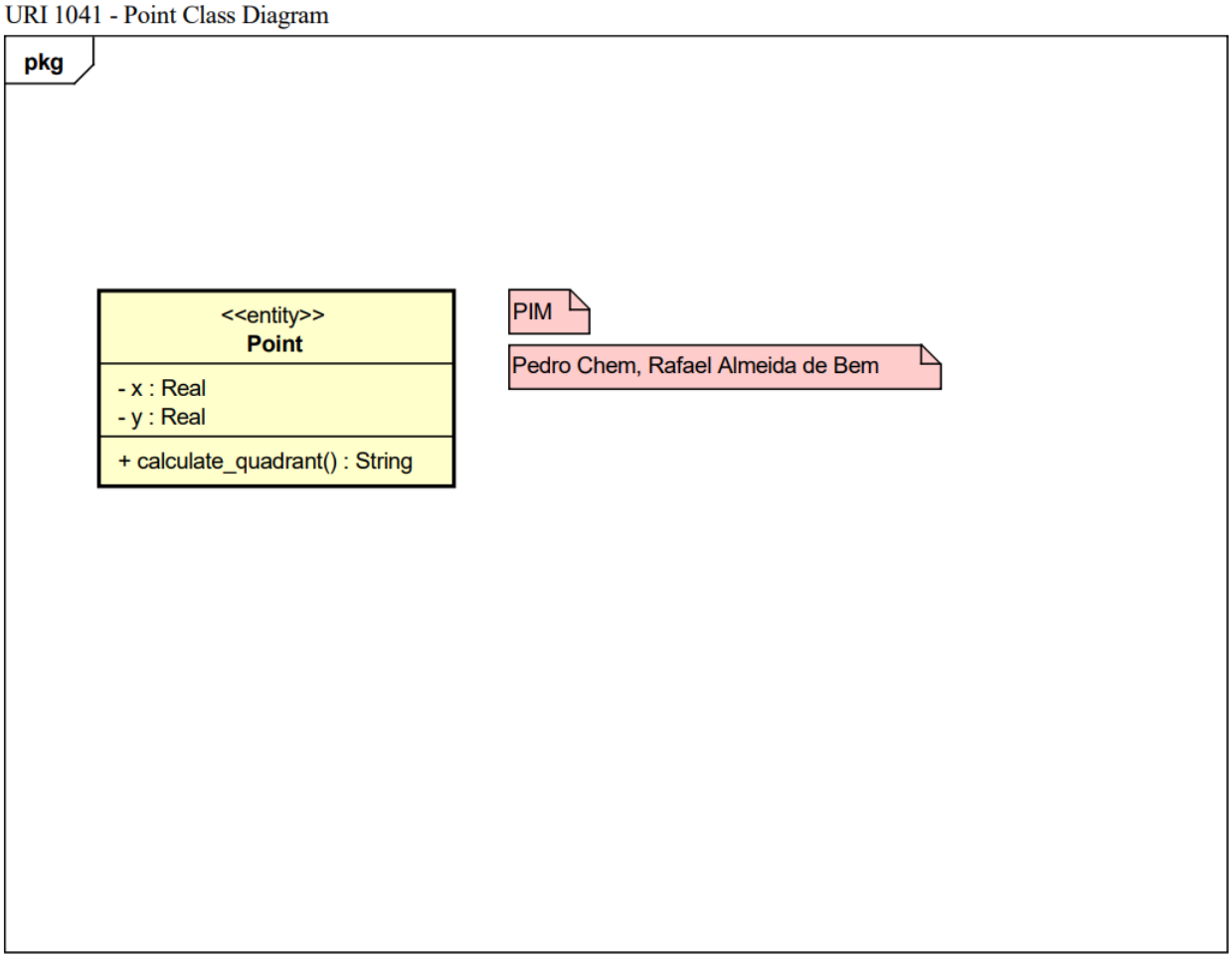

Figura 1.3: Modelo Independente de Plataforma

A Figura 1.3 apresenta o método `Point.calculate_quadrant()` que, a partir das entradas `x` e `y` presentes na instância da classe `Point`, retorna o quadrante correspondente daquele ponto. Se o ponto estiver em (0, 0), o retorno é "Origem".

Modelos

2.1 Modelo Inclusivo

Um modelo inclusivo é um modelo que foi esboçado rapidamente utilizando ferramentas simples como um quadro branco ou cartões [1]. Um exemplo desse modelo é o diagrama de casos de uso.

2.2 Modelo Independente de Plataforma

Um modelo independente de plataforma, ou PIM (*Platform Independent Model*), é uma descrição das funcionalidades de um sistema independentemente das características de implementação em plataformas específicas [2]. Os tipos (*String*, *Bool*, *Inteiro*, etc.) em um PIM são abstrações dos tipos implementados em cada plataforma.

Lista de Verificação

Para contribuir com a qualidade deste trabalho, foi elaborado uma lista de validação. A lista é composta por 7 itens que tem o propósito de validar que os modelos e demais outras entregas estão corretas.

1. O nome dos participantes está visível no diagrama. ✓
2. O diagrama de casos de uso apresenta, no mínimo, um ator. ✓
3. O diagrama independente de plataforma apresenta apenas tipos primitivos independentes de plataforma. ✓
4. O diagrama de casos de uso apresenta a referência ao site onde o enunciado de programação foi retirado. ✓
5. O diagrama independente de plataforma apresenta corretamente o(s) atributo(s) e o(s) método(s) necessário(s). ✓
6. O trabalho apresenta referências às fontes consultadas. ✓
7. O trabalho está disponível em um repositório. ✓

Conclusão

Neste trabalho foram apresentados dois modelos com o propósito de modelar um enunciado de programação retirado do site URI Online Judge [3]. Além disso, foi elaborado uma lista de verificação para controlar a qualidade do trabalho.

Referências Bibliográficas

[1] Scott W. Ambler. Approaches to agile model driven development (amdd). `http://agilemodeling.com/essays/amddApproaches.htm`. Acessado: 29-08-2020. 5

[2] Leopoldo Vinicio Venegas Loor. Arquitectura manejada por modelos. *Revista San Gregorio*, 2014. 5

[3] Neilor Tonin. Uri online judge. `https://www.urionlinejudge.com.br`, Agosto 2020. 2, 7