



**UNIVERSIDADE FEDERAL
DE OURO PRETO**

TRABALHO FINAL

**Sprint 1 - Desenvolvimento de um Framework C++ para a
Construção de Simulações Baseadas na Dinâmica de Sistemas**

ENYA LUÍSA GOMES DOS SANTOS
19.2.4201

Relatório apresentado por exigência da disciplina
BCC322 - ENGENHARIA DE SOFTWARE I, da
Universidade Federal de Ouro Preto.
Professor: TIAGO GARCIA DE SENNA CARNEIRO

OURO PRETO - MG
2021

Objetivo

O objetivo principal do projeto se dá pelo desenvolvimento de um Framework C++ para a Construção de Simulações Baseadas na Dinâmica de Sistemas. Porém, o primeiro passo (SPRINT 1) consiste em entender o produto e organizá-lo para iniciar seu desenvolvimento.

Casos de uso

Para exemplificar os casos, um fluxo (flow) pode ou não conter um sistema (system) com de entrada e outro como de saída. E com esse conceito foi pensado oito situações de uso, essas estão listadas e ilustradas abaixo.

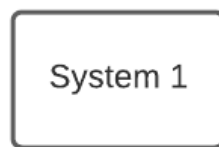


Figura 1 - Sistema sem fluxo



Figura 2 - Fluxo em sistema



Figura 3 - Um sistema com um fluxo com saída.

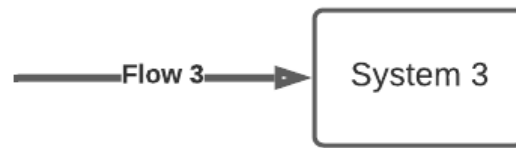


Figura 4 - Um sistema com um fluxo de entrada.



Figura 5 - Um fluxo com um sistema de entrada e um de saída

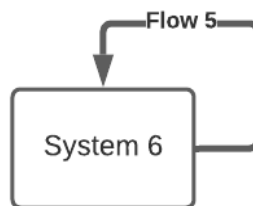


Figura 6 - Um sistema com um fluxo conectado a si mesmo

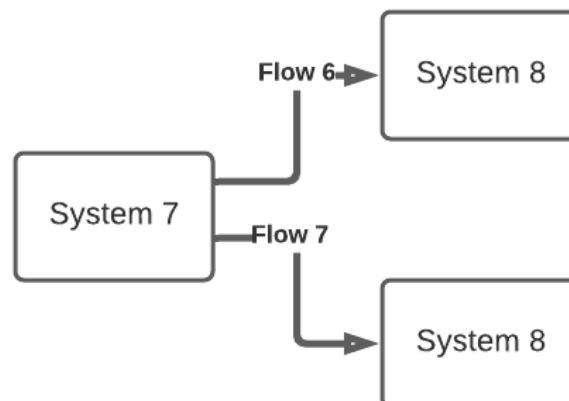


Figura 7 - Um sistema com dois fluxos de saída interligado à outros sistemas como entrada

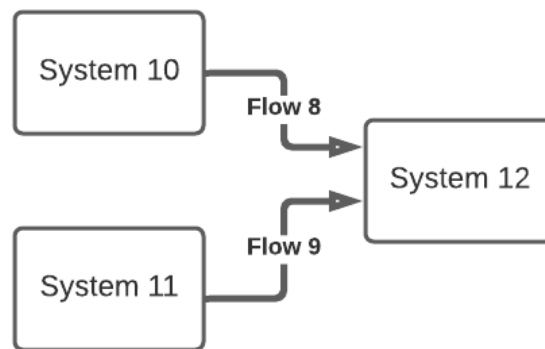


Figura 8 - Um sistema com dois fluxos de entrada interligado à outros sistemas como saída

Critérios de aceitação

Partindo do sistema disponibilizado pelo professor, que simula o cliente, foram destacados os critérios de aceitação abaixo.

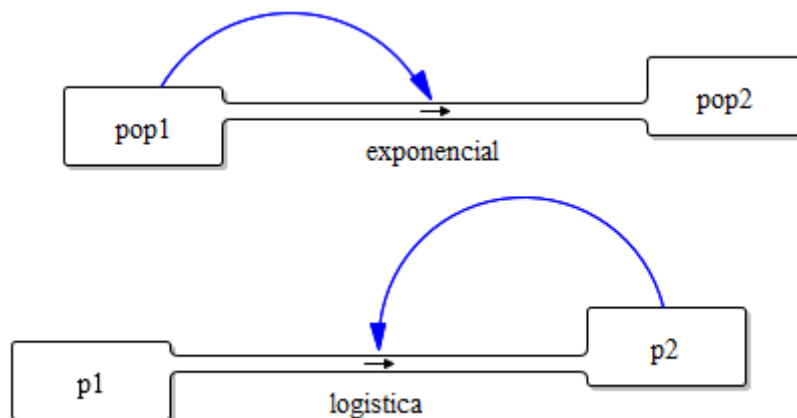


Figura 9 - Relações entre sistemas e fluxos

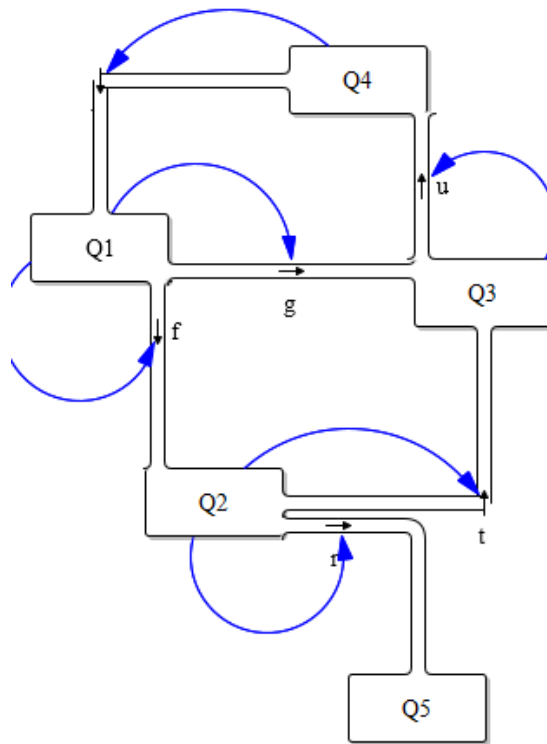


Figura 10 - Fluxo completo para critérios de aceitação

Cenários de teste

Alguns testes referentes a cada um dos casos de uso citados e aos exemplos de critérios de aceitação.

1. Sistema sem fluxo

```
Model m1();
System s1();
m1.add(s1);
m1.run();
```

2. Fluxo sem sistema

```
Model m1();
Flow f1();
m1.add(f1);
m1.run();
```

3. Sistema com um fluxo de saída

```
Model m1();
System s1()
```

```
Flow f1();  
m1.add(s1);  
m1.add(f1);  
f1.setInput(s1);  
f1.setOutput(null);  
m1.run();
```

4. Sistema com um fluxo de entrada

```
Model m1();  
System s1()  
Flow f1();  
m1.add(s1);  
m1.add(f1);  
f1.setInput(null);  
f1.setOutput(s1);  
m1.run();
```

5. Sistemas interligados por um fluxo

```
Model m1();  
System s1();  
System s2();  
Flow f1();  
m1.add(s1);  
m1.add(s2);  
m1.add(f1);  
f1.setInput(s1);  
f1.setOutput(s2);  
m1.run();
```

6. Sistema com um fluxo conectado a si mesmo

```
Model m1();  
System s1()  
Flow f1();  
m1.add(s1);  
m1.add(f1);  
f1.setInput(s1);  
f1.setOutput(s1);  
m1.run();
```

7. Sistema com dois fluxos de saída interligados à dois sistemas como entrada

```
Model m1();
```

```
System s1();
System s2();
System s3();
Flow f1();
Flow f2();
m1.add(s1);
m1.add(s2);
m1.add(s3);
m1.add(f1);
m1.add(f2);
f1.setInput(s1);
f1.setOutput(s2);
f2.setInput(s1);
f3.setOutput(s3);
m1.run();
```

8. Sistemas com dois fluxos de entrada interligados à dois sistemas como saída

```
Model m1();
System s1();
System s2();
System s3();
Flow f1();
Flow f2();
m1.add(s1);
m1.add(s2);
m1.add(s3);
m1.add(f1);
m1.add(f2);
f1.setInput(s1);
f1.setOutput(s3);
f2.setInput(s2);
f3.setOutput(s3);
m1.run();
```

9. Sistema exponencial do Vensim [Critério de aceitação]

```
Model m1();
System pop1();
System pop2();
Flow exponencial();
m1.add(pop1);
m1.add(pop2);
```

```
m1.add(exponencial);  
m1.setInitialTime(0);  
m1.setFinalTime(100);  
pop1.setValue(100);  
exponencial.setInput(s1);  
exponencial.setOutput(s2);  
exponencial.setEquation("0.01 * s1");  
m1.run();  
assert(c1.getFinalValue() == 63.3968);
```

10. Sistema logística do Vensim [Critério de aceitação]

```
Model m1();  
System p1();  
System p2();  
Flow logistica();  
Number Pmax = 70;  
Number rate = 0.01;  
m1.add(p1);  
m1.add(p2);  
m1.add(logistica);  
m1.setInitialTime(0);  
m1.setFinalTime(100);  
p1.setValue(100);  
p2.setValue(10);  
logistica.setInput(s1);  
logistica.setOutput(s2);  
logistica.setEquation("0.01 * s2", Pmax);  
m1.run();  
assert(m1.getFinalValue() == 21.833);
```


UML

UML para o projeto da API

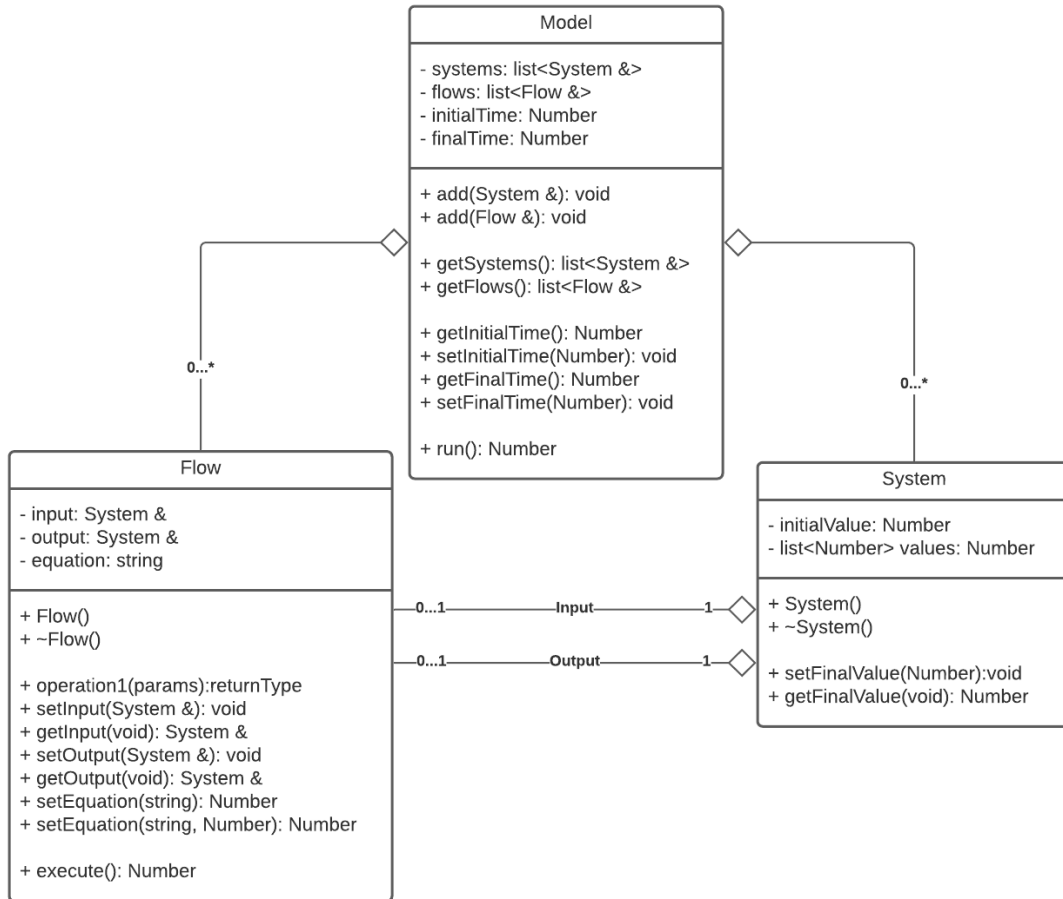


Figura 11 - Diagrama UML para o projeto da API