UNIVERSIDADE FEDERAL DE OURO PRETO
CAMPUS OURO PRETO - OURO PRETO
CIÊNCIA DA COMPUTAÇÃO - 23.2
DISCIPLINA BCC 264

Felipe Braz Marques – 22.1.4030

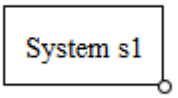
Sprint 1 – Teoria dos Sistemas

Casos de uso

Caso 1: System vazio

Tipo 1: construtor
System s1(double initialValue = 0);

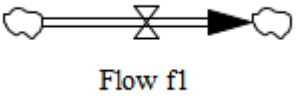
Tipo 2: setando o valor
System s1();
s1.setValue(double initialValue = 0);



Caso 2: Flow sozinho

Tipo 1: construtor
Flow f1(Element *origin = null, Element *destiny = null);

Tipo 2: setando os valores
Flow f1();
f1.setOrigin(Element *origin = null);
f1.setDestiny(Element *destiny = null);



Caso 3: Variable sozinha

Tipo 1: construtor
Variable v1(double initialValue = 0, Element *out = null);

Tipo 2: setando os valores
Variable v1();
v1.setInitialValue(double initialValue = 0);
v1.setDestiny(Element *out = null);

Caso 4: Model vazio

Tipo 1: construtor
Model c1(Element *elements = null);

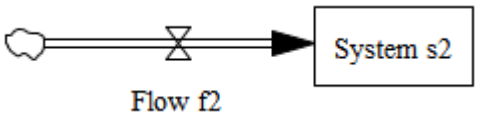
Tipo 2: setando os Elements
Model c1();
c1.addElement(Element *elements = null);

Caso 5: System com um Flow

Tipo 1: System com um Flow de entrada

Tipo 1.1: construtor

```
System s2();
Flow f2(null, &s2);
s2.addIn(&f2);
```



Tipo 1.2: setando os ponteiros

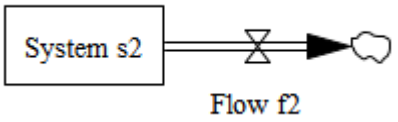
```
System s2();
Flow f2();
f2.setDestiny(&s2);
s2.addIn(&f2);
```

O tipo 1.1 eu considero descartado, pois como discutido em sala de aula, identificar os sentidos do fluxo apenas pela ordem dos parâmetros pode ser de difícil compreensão.

Tipo 2: System com Flow de saída

Tipo 2.1: construtor

```
System s2();
Flow f2(&s2, null);
s2.addOut(&f2);
```



Tipo 2.2: setando os ponteiros

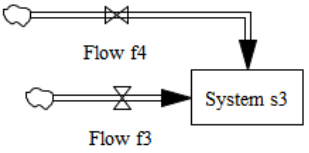
```
System s2();
Flow f2();
f2.setOrigin(&s2);
s2.addOut(&f2);
```

O tipo 1.1 eu considero descartado pelo mesmo motivo citado acima.

Caso 6: System com vários Flows

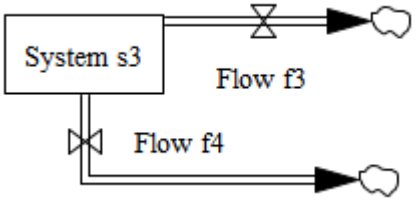
Tipo 1: vários fluxos de entrada

```
System s3()
Flow f3(null, &s3), f4(null, &s3);
s3.addIn(&f3);
s3.addIn(&f4);
```



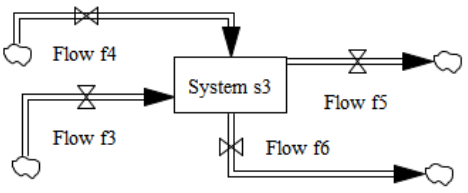
Tipo2 : vários fluxos de saída

```
System s3()
Flow f3(&s3, null), f4(&s3, null);
s3.addOut(&f3);
s3.addOut(&f4);
```



Tipo 3: várias entradas e várias saídas

```
System s3()
Flow f3(&s3, null), f4(&s3, null), f5(null, &s3), f6(null, &s3);
s3.addOut(&f3);
s3.addOut(&f4);
s3.addIn(&f5);
s3.addIn(&f6);
```



Caso 7: System com um Variable

Tipo 1: construtor

```
System s3();
Variable v2(0, &s3);
s3.addIn(&v2);
```

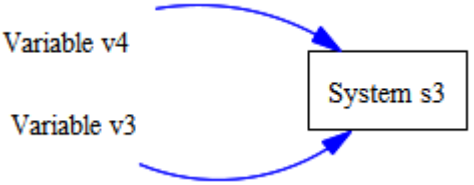
Tipo 2: setando os ponteiros

```
System s3();
Variable v2();
v2.setDestiny(&s3);
s3.addIn(&v2);
```

Caso 8: System com varias Variables

Tipo 1: vários Variables de entrada

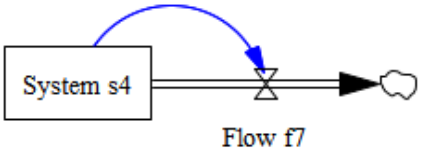
```
System s3()
Variable v3(0, &s3), v4(0, &s3);
s3.addIn(&v3);
s3.addIn(&v4);
```



Caso 9: Elements aponta para um Elements

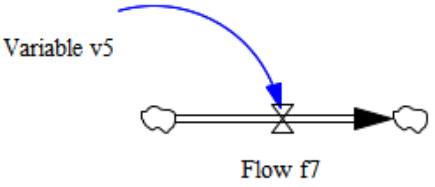
Tipo 1: System aponta para um Flow

```
System s4()
Flow f7()
f7.addIn(&s4)
```



Tipo 2: Variable aponta para um Flow

```
Variable v5()
Flow f7()
f7.addIn(&v5)
```



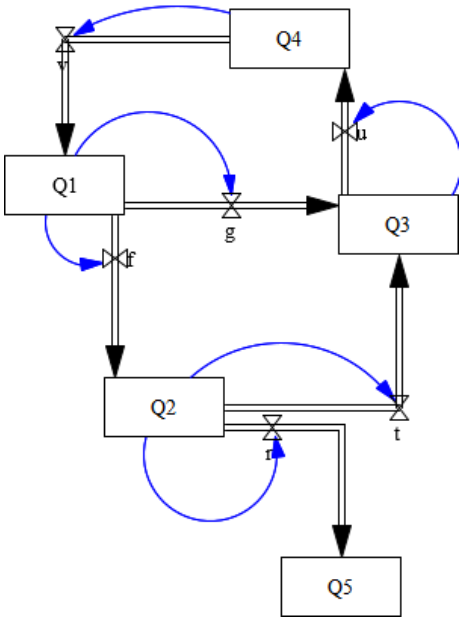
Caso 10: remover Elements

Tipo 1 : removendo de System

```
System s5()
Flow f8(null, &s5), f9(&s5, null);
s5.removeIn(f8);
s5.removeOut(f9)
```

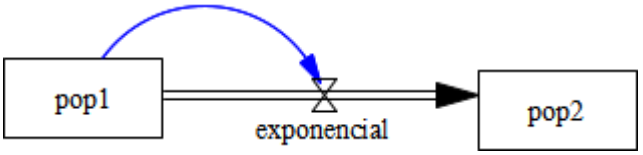
Critério de aceitação

```
Model m();
System Q1(100), Q2(0), Q3(100), Q4(0), Q5(0);
Flow f(&Q1, &Q2), r(&Q2, &Q5), t(&Q5, &Q3), g(&Q1, &Q3), u(&Q3, &Q4),v(&Q4 &Q1);

m.addElement(&Q1);
m.addElement(&Q2);
m.addElement(&Q3);
m.addElement(&Q4);
m.addElement(&Q5);
m.addElement(&f);
m.addElement(&r);
m.addElement(&t);
m.addElement(&g);
m.addElement(&u);
m.addElement(&v);

v.addIn(&Q4);
u.addIn(&Q3);
g.addIn(&Q1);
f.addIn(&Q1);
t.addIn(&Q2);
r.addIn(&Q2);

cout << m.runModel() << endl;
```

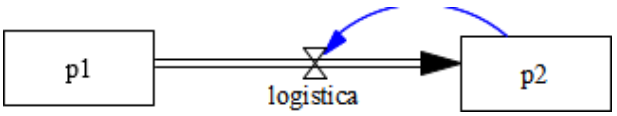


Cenário de teste

```
Model exp();
System pop1(100), pop2(0);
Flow exponencial(&pop1, &pop2);

exponencial.addIn(&pop1);

exponcial.execute(function(Element *origin, Element *destiny, vector<Element> *){return 0.01
* this.in[]->getInitialValue()});
exp.addElement(&pop1, &pop2);
exp.addElement(&exponencial);

cout << exp.runModel() << endl;
```




```
Model log();
System p1(100), p2(10);
Flow logistica(&p1, &p2);

logistica.addIn(&p2);
logistica.execute(function(Element *origin, Element *destiny, vector<Element> *){return 0.01
* this.in[0]->getInitialValue() * (1 – this.in[0]->getInitialValue()/70)});
log.addElement(&p1, &p2);
log.addElement(&logistica);
```



log.runModel()

UML