Caio Silas de Araujo Amaro
21.1.4111

Sprint 1
Engenharia de software

Universidade Federal de Ouro Preto
Maio de 2023

Como um Fluxo está configurado:

```
class Flow{
    public:
        Flow();
        virtual ~Flow();
        Flow(string name = "",System *source = NULL,System *destiny = NULL){
            set.name(name);
            set.source(*source);
            set.destiny(*destiny);
        }
    protected:
        string name;
        System *source;
        System *destiny;
    private:
};
```

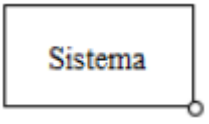
```
class MyFlow : public Flow{
    public:
        Myflow();
        ~Myflow();
        double Execute(void){
            return 0.01*(destine->getValue())*(1-(destine->getValue())/70);
        }
    protected:

    private:
        MyFlow(const string name = "", System *source = NULL, System *destiny = NULL) : Flow(name, source, destiny) {}
}
```

Como utilizar a classe criada:

```
MyFlow flow = new MyFlow(nome,source,destiny);
```

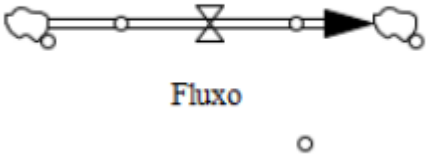
Caso 1: Sistema isolado sem fluxo

```
// Caso de uso 1:
System s1();
s1.setName("System1");
s1.setValue(100);
System s1("System1", 100);
```



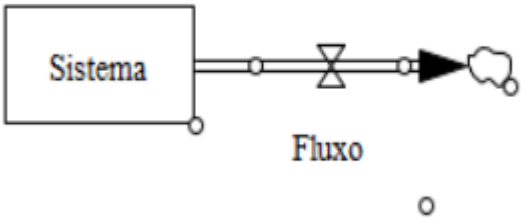
Caso 2: Fluxo em origem e sem destino

```
// Caso de uso 2:
 Flow f1();
 f1.setName("Flow1");
 f1.setSource(NULL);
 f1.setDestine(NULL);
 Flow f1("Flow1", NULL, NULL);
```



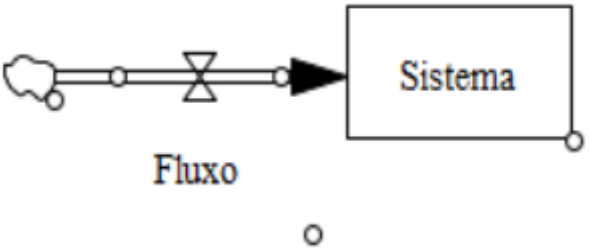
Caso 3:Fluxo com um sistema de origem

```
// Caso de uso 3:
System s1("System1", 100);
Flow f1("Flow1", &s1, NULL);
System s1("System1", 100);
Flow f1();
f1.setName("Flow1");
f1.connect(&s1, NULL);
```



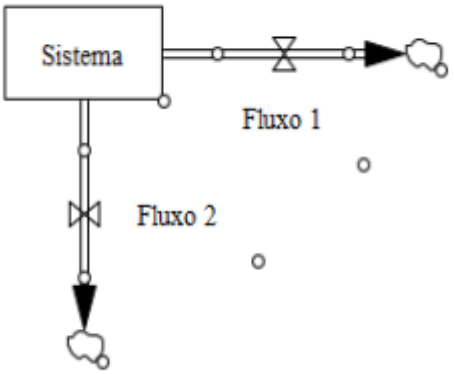
Caso 4: Fluxo com um sistema de destino

```
// Caso de Uso 4:
System s1("System1", 100);
Flow f1("Flow1", NULL, &s1);
System s1("System1", 100);
Flow f1();
f1.setName("Flow1");
f1.connect(NULL, &s1);
```



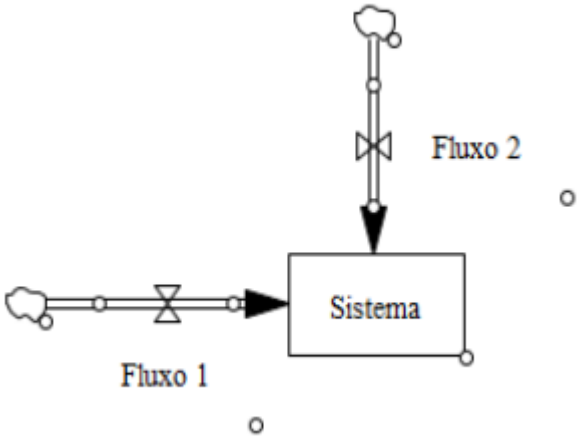
Caso 5:Sistema com dois fluxos de saída

```
// Caso de Uso 5:
System s1("System1", 100);
Flow f1("Flow1", &s1, NULL);
Flow f2("Flow2", &s1, NULL);
System s1("System1", 100);
Flow f1();
f1.setName("Flow1");
f1.connect(&s1, NULL);
Flow f2();
f2.setName("Flow2");
f2.connect(&s1, NULL);
```



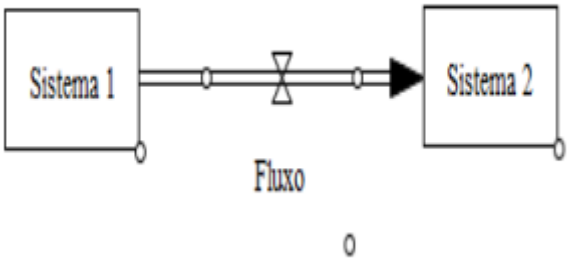
Caso 6:Sistema com dois fluxos de entrada

```
// Caso de Uso 6:
System s1("System1", 100);
Flow f1("Flow1", NULL, &s1);
Flow f2("Flow2", NULL, &s1);
System s1("System1", 100);
Flow f1();
f1.setName("Flow1");
f1.connect(NULL, &s1);
Flow f2();
f2.setName("Flow2");
f2.connect(NULL, &s1);
```



Caso 7:

```
// Caso de Uso 7:
System s1("System1", 100);
System s2("System2", 0);
Flow f1("Flow1", &s1, &s2);
System s1("System1", 100);
System s2("System2", 0);
Flow f1();
f1.setName("Flow1");
f1.connect(&s1, &s2);
```



Caso 8:

```
// Caso de Uso 8:
Model m1("Model1");
Model m1();
m1.setName("Model1");
m1.setTimeStart(0);
m1.setTimeEnd(100);
m1.setTimeVariance(1);
```

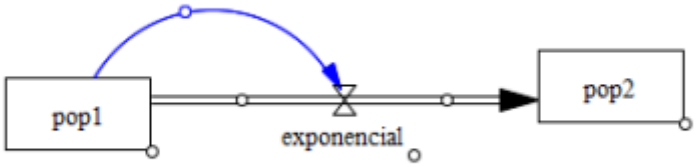
Caso 9:

```
// Caso de uso 9:
System s1("System1", 100);
System s2("System2", 0);
Flow f1("Flow1", &s1, &s2);
Model m1("Model1", 0, 100, 1);
m1.add(&s1);
m1.add(&s2);
m1.add(&f1);
m1.run();
```

Cenários de teste:

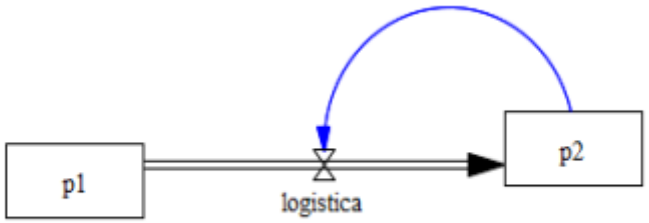
Primeiro Cenário

```
class MyFlow : public Flow{
    public:
        Myflow();
        ~Myflow();
        double Execute(void){
            return 0.01*(destine->getValue())*(1-(destine->getValue())/70);
        }
    protected:

    private:
        MyFlow(const string name = "", System *source = NULL, System *destiny = NULL) : Flow(name, source, destiny) {}
}
    System s1("System1", 100);
    System s2("System2", 0);
    MyFlow f1;
    f1.setName("Flow1");
    f1.connect(&s1, &s2);
    Model m1("Model1");
    m1.add(&s1);
    m1.add(&s2);
    m1.add(&f1);
    m1.run(0, 100, 1);
```



Segundo Cenário

```
class MyFlow : public Flow{
    public:
        Myflow();
        ~Myflow();
        double Execute(void){
            return 0.01*(destine->getValue())*(1-(destine->getValue())/70);
        }
    protected:
    private:
        MyFlow(const string name = "", System *source = NULL, System *destiny = NULL) : Flow(name, source, destiny) {}


}
    System s1("System1", 100);
    System s2("System2", 0);
    MyFlow f1;
    f1.setName("Flow1");
    f1.connect(&s1, &s2);
    Model m1("Model1");
    m1.add(&s1);
    m1.add(&s2);
    m1.add(&f1);
    m1.run(0, 100, 1);
```



Terceiro Cenário

```
class MyFlow : public Flow{
    public:
        Myflow();
        ~Myflow();
        double Execute(void){
            return 0.01*(destine->getValue())*(1-(destine->getValue())/70);
        }
    protected:
    private:
        MyFlow(const string name = "", System *source = NULL, System *destiny = NULL) : Flow(name, source, destiny) {}
}
    System Q1("Q1", 100), Q2("Q2", 0), Q3("Q3", 100), Q4("Q4", 0), Q5("Q5", 0);
    MyFlow f, g, t, r, u, v;
    f.setName("f"); f.connect(&Q1, &Q2);
    g.setName("g"); g.connect(&Q1, &Q3);
    t.setName("t"); t.connect(&Q2, &Q3);
    r.setName("r"); r.connect(&Q2, &Q5);
    u.setName("u"); u.connect(&Q3, &Q4);
    v.setName("v"); v.connect(&Q4, &Q1);
    Model m1("Model1");
    m1.add(&Q1);
    m1.add(&Q2);
    m1.add(&Q3);
    m1.add(&Q4);
    m1.add(&Q5);
    m1.add(&f);
    m1.add(&g);
    m1.add(&t);
    m1.add(&r);
    m1.add(&u);
    m1.add(&v);
    m1.run(0, 100, 1);
```

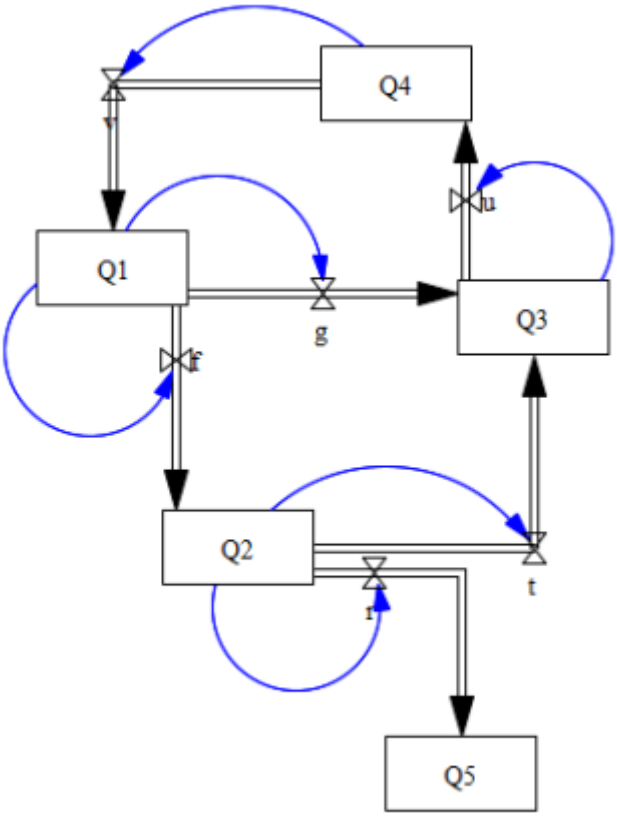
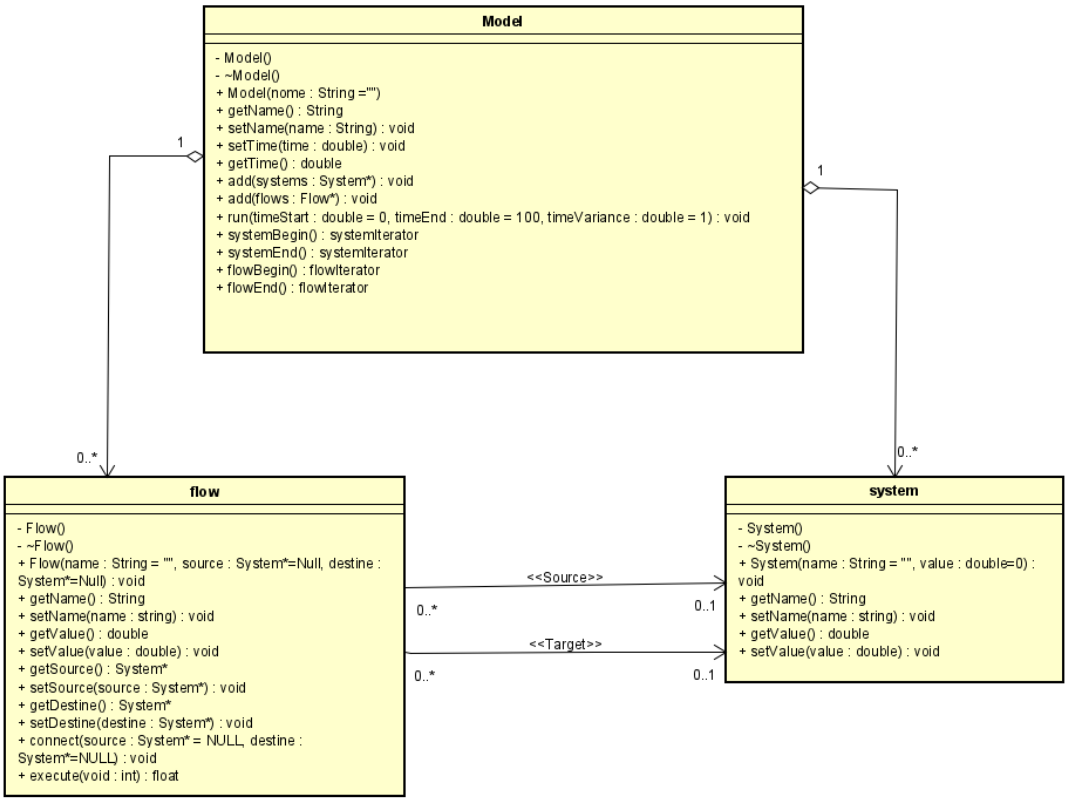


Diagrama uml:


Model
- Model()
- ~Model()
+ Model(nome : String ="")
+ getName() : String
+ setName(name : String) : void
+ setTime(time : double) : void
+ getTime() : double
+ add(systems : System*) : void
+ add(flows : Flow*) : void
+ run(timeStart : double = 0, timeEnd : double = 100, timeVariance : double = 1) : void
+ systemBegin() : systemIterator
+ systemEnd() : systemIterator
+ flowBegin() : flowIterator
+ flowEnd() : flowIterator
1
1
0..*
0..*
flow
- Flow()
- ~Flow()
+ Flow(name : String = "", source : System*=Null, destine : System*=Null) : void
+ getName() : String
+ setName(name : string) : void
+ getValue() : double
+ setValue(value : double) : void
+ getSource() : System*
+ setSource(source : System*) : void
+ getDestine() : System*
+ setDestine(destine : System*) : void
+ connect(source : System* = NULL, destine : System*=NULL) : void
+ execute(void : int) : float
<<Source>>
0..*
0..1
<<Target>>
0..*
0..1
system
- System()
- ~System()
+ System(name : String = "", value : double=0) : void
+ getName() : String
+ setName(name : string) : void
+ getValue() : double
+ setValue(value : double) : void