# PipeSim

Thiago Aramaki

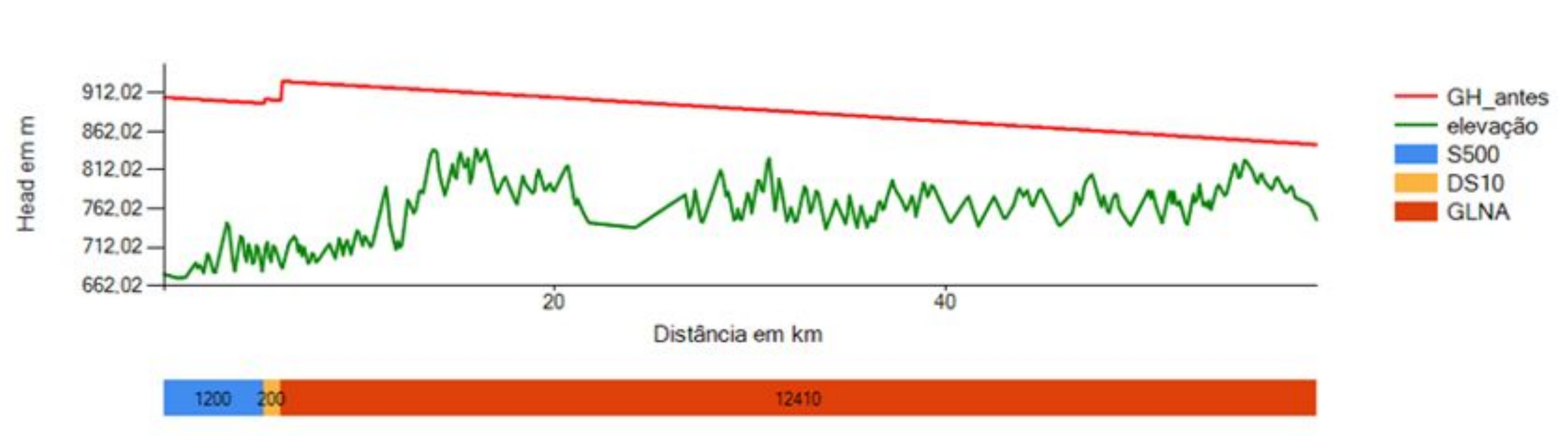
# Sumário

# Domínio escolhido

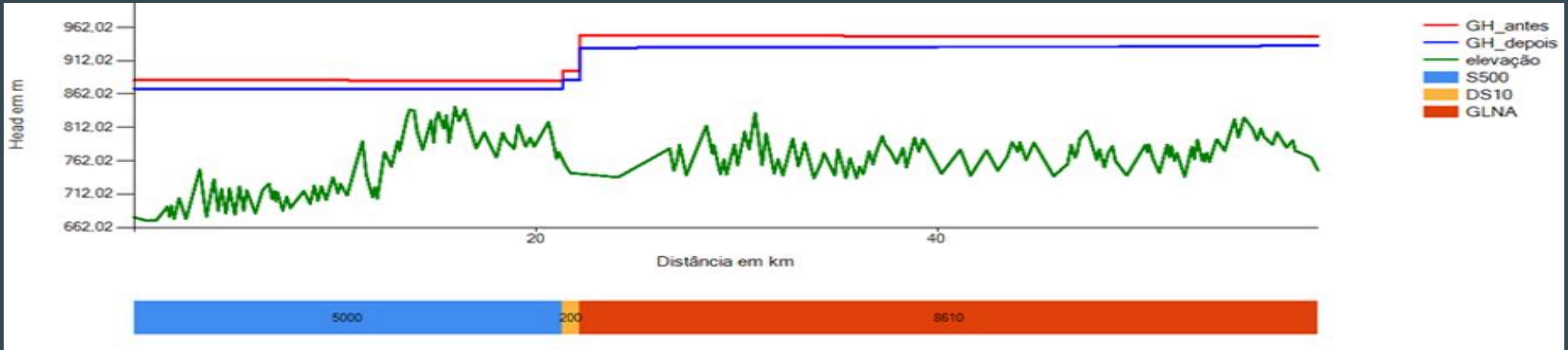
# Operação de oleodutos

# Descrição do problema

# Descrição do problema - Gradiente Hidráulico em operação



# Descrição do problema - Gradiente Hidráulico em repouso



# Domínio escolhido - premissas

- Somente oleodutos, gasodutos não estão contemplados
- Não será realizado o cálculo com a parte térmica inserida
- O modelo deve aceitar polidutos (mais de um produto sendo transportado)
- Quebra de coluna não será  detectada
- Dutos bidirecionais não contemplados
- Gradiente hidráulico com coluna quebrada não será realizado
- Somente o gradiente hidráulico pelo expedidor será realizado
- Somente o regime permanente está contemplado
- Equipamentos, válvulas, e bombas não estão contemplados

# Modelo matemático

1400

$$Re = \frac{\rho \cdot v \cdot D}{\mu}$$

$$f = \frac{64}{Re}$$

$$f = 0.0457$$

2750

$$f = 0.25 \cdot \log_{10}\left(\frac{k}{1000} \cdot \frac{1}{3,7 \cdot D_{int}} + 5,74 \cdot Re^{-0,9}\right)^{-2}$$

# Modelo matemático

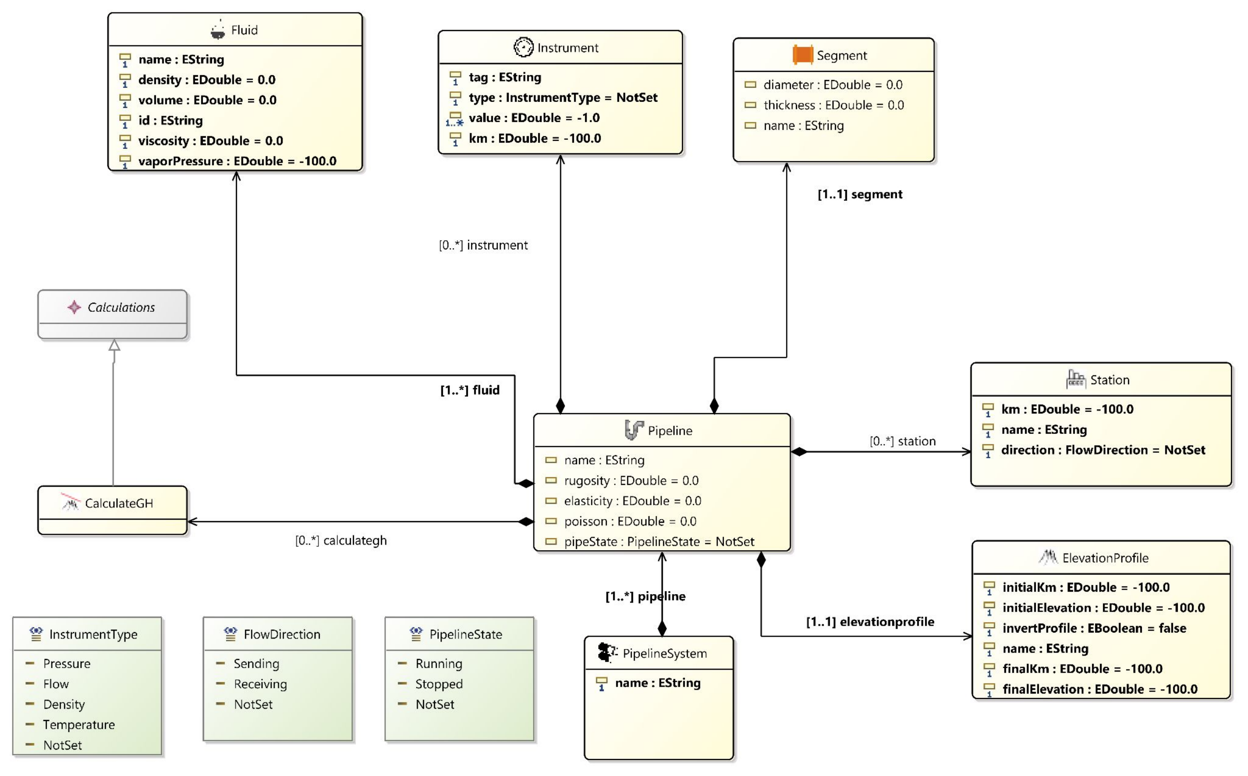
$$H = \frac{10 \cdot P}{d} + Z$$

$$P = \frac{d \cdot (H - Z)}{10}$$

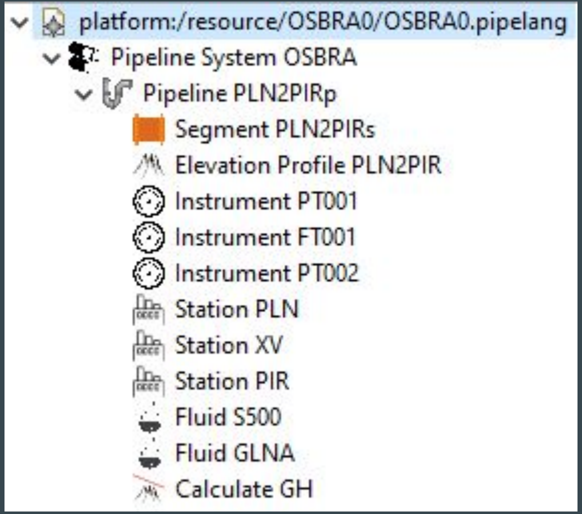
$$h_f = f \cdot \frac{L}{D} \cdot \frac{v^2}{2g}$$

# Descrição do PIM

# Diagrama do PIM



# Modelo de exemplo do PIM




```xml
<?xml version="1.0" encoding="UTF-8"?>
<PipeLang:PipelineSystem xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:PipeLang="http://www.example.org/PipeLang" name="OSBRA">
  <pipeline name="PLN2PIRp" rugosity="0.045" elasticity="1899731.0"
poisson="0.3">
    &lt;segment diameter="20.0" thickness="0.25" name="PLN2PIRs"/>
    <elevationprofile initialKm="0.0" initialElevation="700.0"
name="PLN2PIR" finalKm="99.0" finalElevation="854.0"/>
    <instrument tag="PT001" type="Pressure" km="0.0">
      <value>72.0</value>
    </instrument>
    <instrument tag="FT001" type="Flow" km="0.0">
      <value>900.0</value>
    </instrument>
    <instrument tag="PT002" type="Pressure" km="99.0">
      <value>7.0</value>
    </instrument>
    <station km="0.0" name="PLN" direction="Sending"/>
    <station km="45.0" name="XV" direction="Sending"/>
    <station km="95.0" name="PIR" direction="Receiving"/>
    <fluid name="S500" density="840.0" volume="10000.0" id="2702200001"
viscosity="345.0" vaporPressure="0.0"/>
    <fluid name="GLNA" density="720.0" volume="-1.0" id="2702200001"
viscosity="1.0" vaporPressure="0.0"/>
    <calculategh/>
  </pipeline>
</PipeLang:PipelineSystem>
```

# Modelo PIM centrado em dutos

Properties ☒ | Problems

### Station XV

Main | Semantic | Style | Appearance

| Property | Value |
|---|---|
| ⌄ Station XV | |
| Direction | NotSet |
| Km | 45.0 |
| Name | XV |

### Segment PLN2PIRs

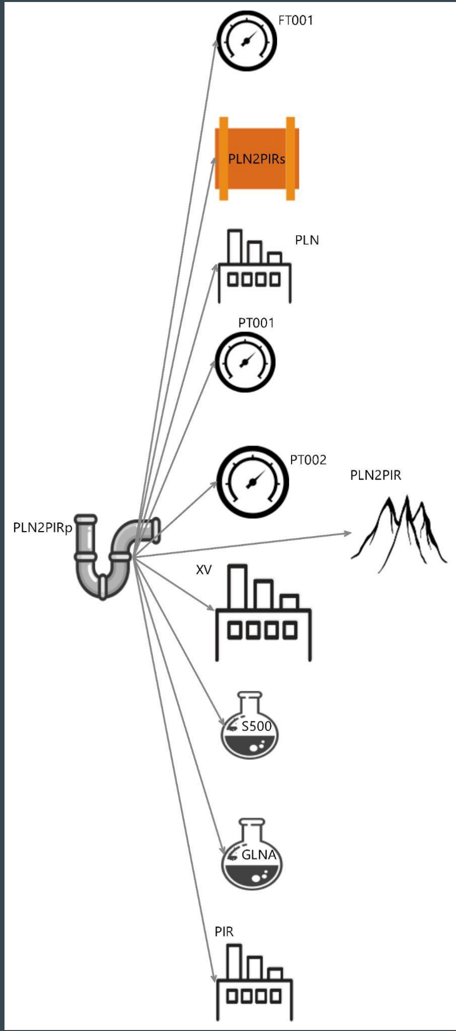
Main | Semantic | Style | Appearance

| Property | Value |
|---|---|
| ⌄ Segment PLN2PIR | |
| Diameter | 20.0 |
| Name | PLN2PIRs |
| Thickness | 0.25 |

### Elevation Profile PLN2PIR

Main | Semantic | Style | Appearance

| Property | Value |
|---|---|
| ⌄ Elevation Profile P | |
| Final Elevation | 854.0 |
| Final Km | 99.0 |
| Initial Elevation | 700.0 |
| Initial Km | 0.0 |
| Invert Profile | false |
| Name | PLN2PIR |




### Instrument FT001

Main | Semantic | Style | Appearance

| Property | Value |
|---|---|
| ⌄ Instrument FT001 | |
| Km | 0.0 |
| Tag | FT001 |
| Type | Flow |
| Value | 0.0 |

### Fluid S500

Main | Semantic | Style | Appearance

| Property | Value |
|---|---|
| ⌄ Fluid S500 | |
| Density | 840.0 |
| Id | 2702200001 |
| Name | S500 |
| Vapor Pressure | 0.0 |
| Viscosity | 1.0 |
| Volume | 10000.0 |

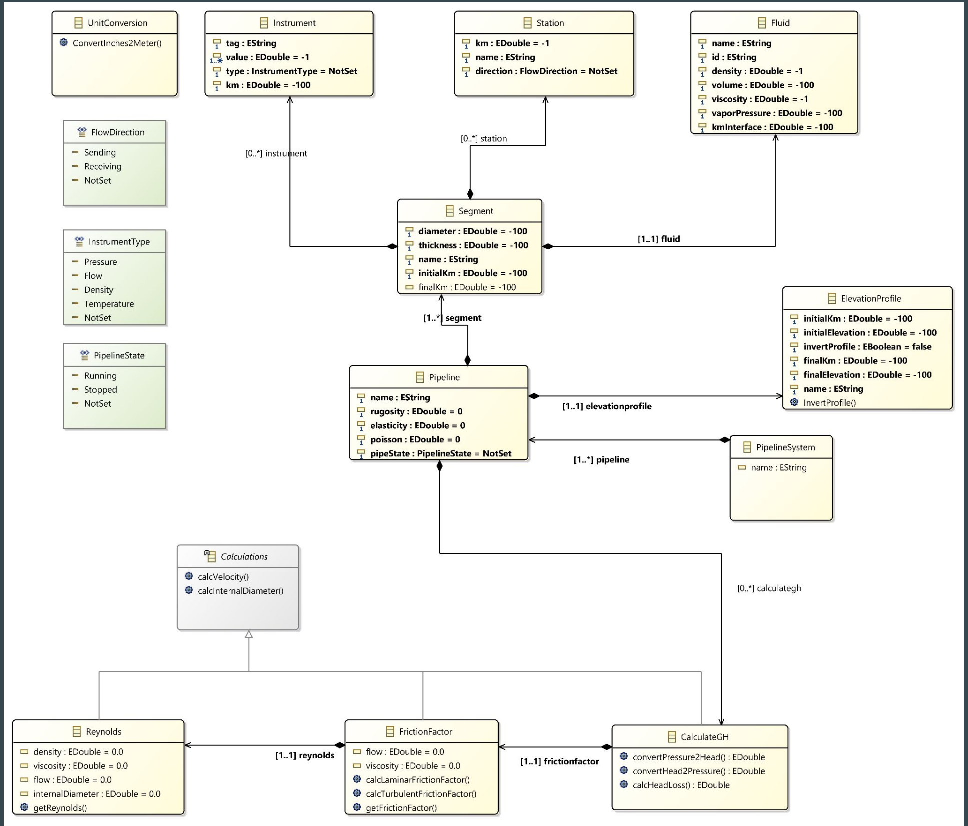
### Pipeline PLN2PIRp

Main | Semantic | Style | Appearance

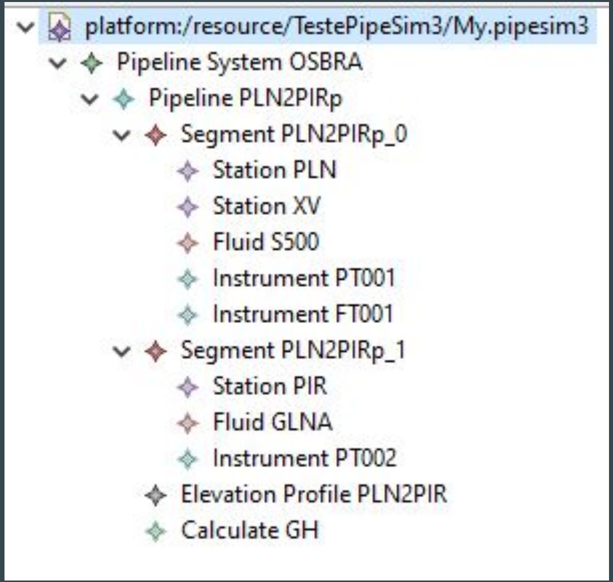
| Property | Value |
|---|---|
| ⌄ Pipeline PLN2PIRp | |
| Elasticity | 1899731.0 |
| Name | PLN2PIRp |
| Pipe State | NotSet |
| Poisson | 0.3 |
| Rugosity | 0.045 |

# Descrição do PSM

# Diagrama do PSM



# Modelo de exemplo do PSM

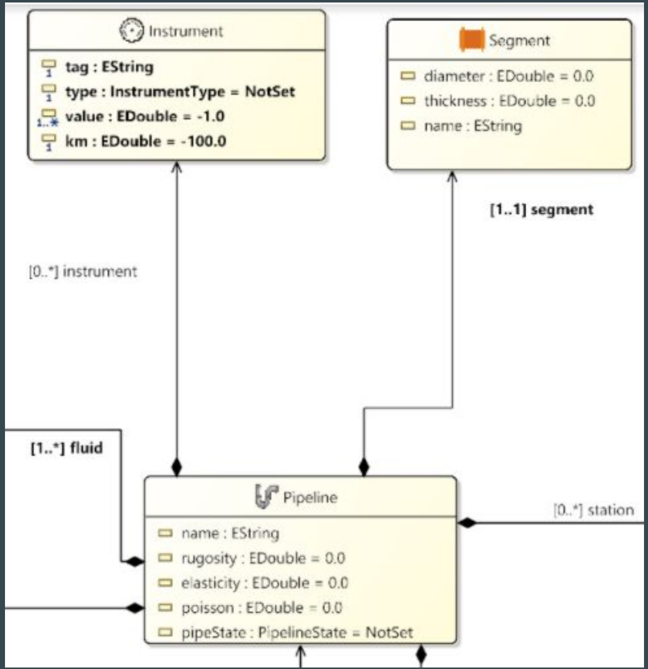




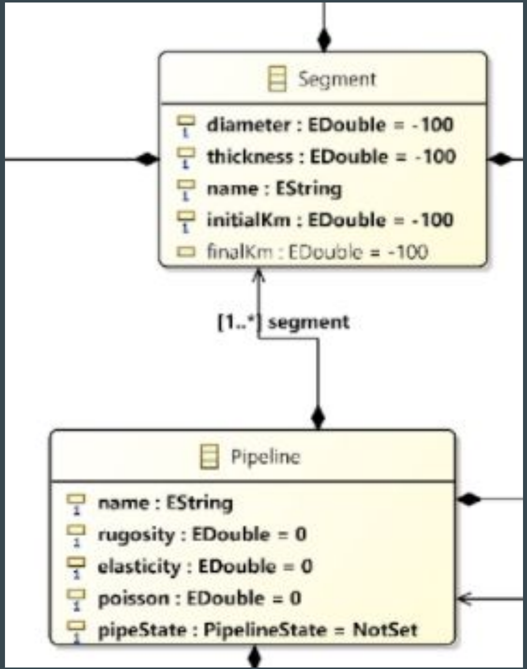
```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<PipeSim3:PipelineSystem xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:PipeSim3="http://www.example.org/PipeSim3" name="OSBRA">
  <pipeline name="PLN2PIRp" rugosity="0.045" elasticity="1899731.0"
poisson="0.3">
    &lt;segment diameter="20.0" thickness="0.25" name="PLN2PIRp_0"
initialKm="0.0" finalKm="52.0">
      <station km="0.0" name="PLN" direction="Sending"/>
      <station km="45.0" name="XV" direction="Sending"/>
      <fluid name="S500" id="2702200001" density="840.0" volume="10000.0"
viscosity="345.0" vaporPressure="0.0" kmInterface="52.0"/>
      <instrument tag="PT001" type="Pressure" km="0.0">
        <value>72.0</value>
      </instrument>
      <instrument tag="FT001" type="Flow" km="0.0">
        <value>900.0</value>
      </instrument>
    &lt;/segment&gt;
    &lt;segment diameter="20.0" thickness="0.25" name="PLN2PIRp_1"
initialKm="52.0" finalKm="99.0">
      <station km="95.0" name="PIR" direction="Receiving"/>
      <fluid name="GLNA" id="2702200001" density="720.0" volume="9075.0"
viscosity="1.0" vaporPressure="0.0" kmInterface="99.0"/>
      <instrument tag="PT002" type="Pressure" km="99.0">
        <value>7.0</value>
      </instrument>
    &lt;/segment&gt;
    <elevationprofile initialKm="0.0" initialElevation="700.0"
finalKm="99.0" finalElevation="854.0" name="PLN2PIR"/>
    <calculategh/>
  </pipeline>
</PipeSim3:PipelineSystem>
```

# Diferenças => Pipeline / Segments

PIM

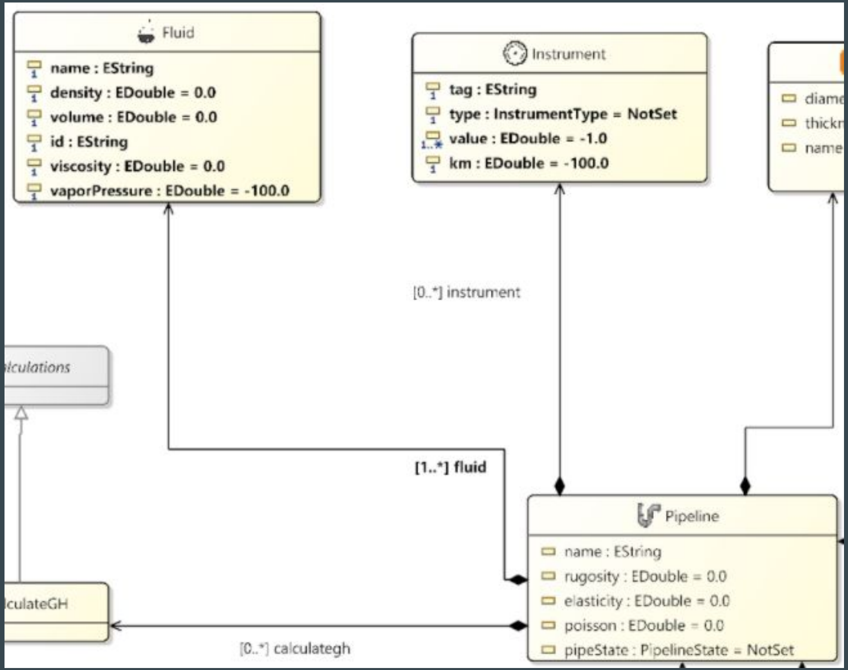



PSM

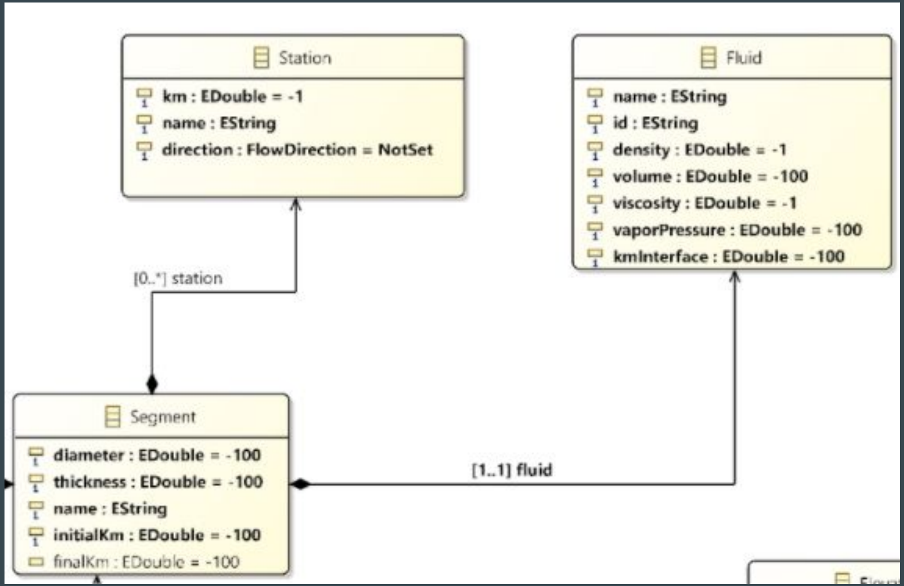


# Diferenças => Fluid

## PIM



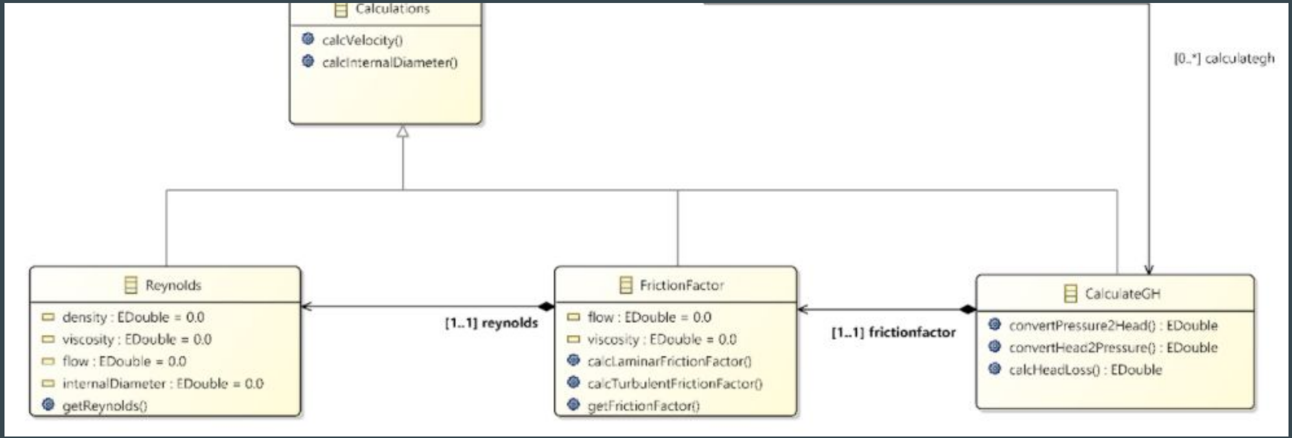

## PSM

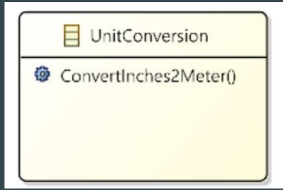


# Diferenças => Classes auxiliares

PIM

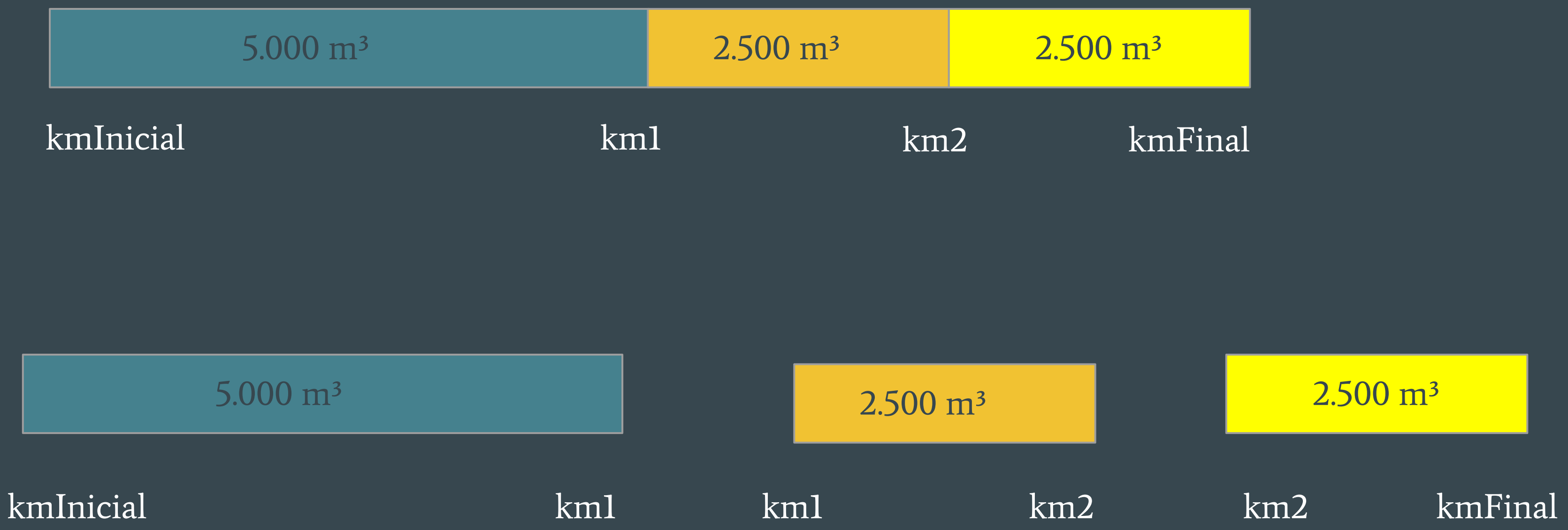
PSM



# M2M -> ATL

# ATL - Cálculo do volume do segmento baseado no volume dos fluidos



# ATL - Cálculo do volume do pipeline

```
helper def: comprimento(kmInicial: Real, kmFinal: Real): Real =
      (kmFinal - kmInicial) * 1000;

helper def: diametroInterno(diametroExterno: Real, espessura: Real): Real =
      (diametroExterno - 2 * espessura) * 0.0254;

helper def: area(diametroExterno: Real, espessura: Real): Real =
      180.toRadians() * thisModule.diametroInterno(diametroExterno,
espessura) * thisModule.diametroInterno(diametroExterno, espessura)  / 4;

helper def: calcVolume(diametroExterno: Real, espessura: Real, kmInicial:
Real, kmFinal:
            Real): Real =
      thisModule.area(diametroExterno, espessura) *
thisModule.comprimento(kmInicial,
                  kmFinal);

helper def: totalVolume(pipe: PipeLang!Pipeline): Real =
            thisModule.calcVolume(pipe.segment.diameter,
pipe.segment.thickness, pipe.elevationprofile.initialKm ,
pipe.elevationprofile.finalKm);
```

# ATL - Cálculo do volume do segmento baseado no volume dos fluidos

```
helper context PipeLang!Fluid def:calcKMInterface() : Real =
	if thisModule.numFluids=1 then
		thisModule.totalVolume(self.refImmediateComposite())
	else
		if thisModule.numFluids>1 then
			PipeLang!Fluid.allInstances().asSequence().subSequence(1,
PipeLang!Fluid.allInstances().asSequence().indexOf(self))->iterate(e; soma:
Real = 0 |
			if e.volume <> -1.0 then
				(soma +
e.volume)/thisModule.totalVolume(self.refImmediateComposite()) *
self.refImmediateComposite().elevationprofile.finalKm
			else

self.refImmediateComposite().elevationprofile.finalKm
			endif
		)
		else
			self.refImmediateComposite().elevationprofile.finalKm
		endif
	endif;
```

# ATL - Setar o initialKm do segment

```
helper context PipeLang!Fluid def: getInitialKm(): Real =
      let index : Integer =
PipeLang!Fluid.allInstances().asSequence().indexOf(self) in
            if index = 1 then
                  self.refImmediateComposite().elevationprofile.initialKm

            else

PipeSim!Fluid.allInstances().asSequence().at(index-1).kmInterface
            endif;
```

# Identificação do posicionamento das estações e instrumentos

```
helper def: getStation(firstKm:Real, lastKm:Real):
Sequence(PipeLang!Station) =
      PipeLang!Station.allInstances()->iterate(st; conjunto:
Sequence(PipeLang!Station) = Sequence{} |
      if st.km >= firstKm and st.km <= lastKm then
            conjunto->append(st)
      else
            conjunto
      endif

);

helper def: getInstrument(firstKm:Real, lastKm:Real):
Sequence(PipeLang!Instrument) =
      PipeLang!Instrument.allInstances()->iterate(inst; conjunto:
Sequence(PipeLang!Instrument) = Sequence{} |
      if inst.km >= firstKm and inst.km <= lastKm then
            conjunto->append(inst)
      else
            conjunto
      endif

);
```

# Identificação do posicionamento dos segmentos

```
helper def: getSegments(pipe:PipeLang!Pipeline) : Sequence(PipeSim!Segment)
=
      let firstKm : Real = pipe.elevationprofile.initialKm in
      let lastKm : Real = pipe.elevationprofile.finalKm in
      PipeSim!Segment.allInstances()->iterate(seg; conjunto:
Sequence(PipeLang!Segment) = Sequence{} |
      if seg.initialKm >= firstKm and seg.finalKm <= lastKm then
            conjunto->append(seg)
      else
            conjunto
      endif
);
```

# Mapeamento das regras

```
rule PipelineSystem2PipelineSystem {
	from
		s: PipeLang!PipelineSystem
	to
		t: PipeSim!PipelineSystem (
			name <- s.name,
			pipeline <- s.pipeline
		)
}
-- fazer o mapeamento das classes que são iguais
rule Station2Station {
	from
		s: PipeLang!Station
	to
		t: PipeSim!Station (
			name <- s.name,
			km <- s.km,
			direction <- s.direction
		)
}

rule Instrument2Instrument {
	from
		s: PipeLang!Instrument
	to
		t: PipeSim!Instrument (
			tag <- s.tag,
			value <- s.value,
			type <- s.type,
			km <- s.km
		)
}
```

```
rule Fluid2Fluid{
	from
		flu: PipeLang!Fluid
	to
		t1: PipeSim!Fluid (
			name <- flu.name,
			density <- flu.density,
			id <- flu.id,
			viscosity <- flu.viscosity,
			vaporPressure <- flu.vaporPressure,
			volume <- flu.fluidVolume().round(),
			kmInterface <- flu.calcKMInterface().round()
		),
		seg:PipeSim!Segment (
			name <- flu.refImmediateComposite().name  + '_' +
thisModule.idSeq,
			diameter <- flu.refImmediateComposite().segment.diameter,
			thickness  <-
flu.refImmediateComposite().segment.thickness,
			initialKm <- flu.getInitialKm(),
			finalKm <- flu.calcKMInterface().round(),
			fluid <- flu,
			station <- thisModule.getStation(seg.initialKm,
seg.finalKm),
			instrument <- thisModule.getInstrument(seg.initialKm,
seg.finalKm)

		)
		do {
			thisModule.idSeq <- thisModule.idSeq + 1 ;

			thisModule.println(thisModule.idSeq);
		}
}
```

# Mapeamento do pipeline

```
rule calculateGH {
	from
		s: PipeLang!CalculateGH
	to
		t : PipeSim!CalculateGH (

		)
}
```

```
rule Elevation2Elevation {
	from
		s: PipeLang!ElevationProfile
	to
		t: PipeSim!ElevationProfile (
			name <- s.name,
			initialKm <- s.initialKm,
			initialElevation <- s.initialElevation,
			finalKm <- s.finalKm,
			finalElevation <- s.finalElevation,
			invertProfile <- s.invertProfile
		)
}
```

```
abstract rule Pipeline2Pipeline {
	from
		s: PipeLang!Pipeline
	to
		t: PipeSim!Pipeline (
			name <- s.name,
			rugosity <- s.rugosity,
			elasticity <- s.elasticity,
			poisson <- s.poisson,
			pipeState <- s.pipeState,
			elevationprofile <- s.elevationprofile


		)
}

rule PipelineSegments extends Pipeline2Pipeline{
	from
		s: PipeLang!Pipeline
	to
		t : PipeSim!Pipeline (
			segment <- thisModule.getSegments(s),
			calculategh <- s.calculategh
		)
}
```

# M2T

# Acceleo - Geração das demais classes a partir da raiz

```
[template public generatePipelineSystem(aPipelineSystem : PipelineSystem)]
[comment @main/]

[UnitConversion('UnitConversion')/]
[generateInicializaGeral(aPipelineSystem)/]

[comment para cada sistema podem haver um ou mais pipelines/]
[for (p : Pipeline | aPipelineSystem.pipeline)]
      [generatePipeline(p)/]
      [for (s : Segment | p.segment)]
            [generateSegment(s)/]
            [for (st : Station | s.station)]
                  [generateStations(st)/]
            [/for]
            [for (inst : Instrument | s.instrument)]
                  [generateInstruments(inst)/]
            [/for]
            [for (f : Fluid | s.fluid)]
                  [generateFluids(f)/]
            [/for]
      [/for]
      [for (e : ElevationProfile | p.elevationprofile)]
            [generateElevationProfile(e)/]
      [/for]
[/for]
```

# Acceleo - PipelineSystem

```
[comment na versão atual do PSM, somente um pipelineSystem é possível/]
[file (aPipelineSystem.name + '.vb', false, 'UTF-8')]

Public Class [aPipelineSystem.name/]
[comment para cada sistema, podem existir mais de um pipeline /]

    Public name as String = "[aPipelineSystem.name/]"
    [for (pipe : String | aPipelineSystem.pipeline.name)]
    Public [pipe/] as New [pipe/]
    [/for]
    Public listPipeline as New List(of Object)({ [for (aPipelineSystem.pipeline.name) separator(', ')][self/][/for] })

    Public Sub New()

    End Sub

    Public Sub New(name as String)
        Me.name = name
    End Sub

End Class
[/file]
[/template]
```

# Pipeline com e sem GH

```
[query public hasGH(pipe:Pipeline) : Boolean =
	not pipe.calculategh->isEmpty()
/]
```

```
[template public generatePipeline(aPipe:Pipeline) ? (not hasGH(aPipe)) post
(trim())]

[file (aPipe.name + '.vb', false, 'UTF-8')]

Public Class [aPipe.name/]

	Public name as String = "[aPipe.name/]"
	Public elasticity = [aPipe.elasticity/]
	Public poisson = [aPipe.poisson/]
	Public rugosity = [aPipe.rugosity/]
	[for (seg : String | aPipe.segment.name)]
	Public [seg/] As  New[seg/]
	[/for]
	Public listSegments As New List(of Object)({ [for
(aPipe.segment.name) separator(', ')][self/][/for] })
	Public elevationProfile As [aPipe.elevationprofile.name/]

	Public Sub New()

	End Sub

	Public Sub New(name as String,
				elasticity as Double,
			poisson as Double,
				rugosity as Double,
			listSegments as List(of Object),
				elevationProfile as Object)

		Me.name = name
		Me.elasticity = elasticity
		Me.poisson = poisson
		Me.rugosity = rugosity
		Me.listSegments = listSegments
		Me.elevationProfile = elevationProfile

	End Sub

End Class
[/file]
[/template]
```

```
[template public generatePipeline(aPipe:Pipeline) ? ( hasGH(aPipe)) post
(trim())]

[file (aPipe.name + '.vb', false, 'UTF-8')]
Public Class [aPipe.name/]

	Public name as String = "[aPipe.name/]"
	Public elasticity = [aPipe.elasticity/]
	Public poisson = [aPipe.poisson/]
	Public rugosity = [aPipe.rugosity/]
	[for (seg : String | aPipe.segment.name)]
	Public [seg/] As New [seg/]
	[/for]
	Public listSegments As New List(of Object)({ [for
(aPipe.segment.name) separator(', ')][self/][/for] })
	Public elevationProfile As New [aPipe.elevationprofile.name/]
	Public f As new FrictionFactor

	Public Sub New()

	End Sub
	Public Sub New(name as String,
				elasticity as Double,
			poisson as Double,
				rugosity as Double,
			listSegments as List(of Object),
				elevationProfile as Object)

		Me.name = name
		Me.elasticity = elasticity
		Me.poisson = poisson
		Me.rugosity = rugosity
		Me.listSegments = listSegments
		Me.elevationProfile = elevationProfile
	End Sub
End Class
[/file]
[if (hasGH(aPipe))]
	[generateReynolds('Reynolds')/]
	[generateCalculations('Calculations')/]
	[generateFrictionFactor('FrictionFactor')/]
	[generateCalculateGH('CalculateGH',getFirstPipelineSystem(aPipe))/]
[/if]

[/template]
```

# Geração do segment

```
[template public generateSegment(aSegment:Segment) post (trim())]

[file (aSegment.name + '.vb', false, 'UTF-8')]

Public Class [aSegment.name/]

      Public name as String = "[aSegment.name/]"
      Public diameter = [aSegment.diameter/]
      Public thickness = [aSegment.thickness/]
      Public firstKm = [aSegment.initialKm/]
      Public lastKm = [aSegment.finalKm/]
      [for (seg : String | aSegment.station.name)]
      Public [seg/] As New [seg/]
      [/for]
      [for (seg : String | aSegment.instrument.tag)]
      Public [seg/] As New [seg/]
      [/for]
      Public listStation As New List(of Object)({ [for
(aSegment.station.name) separator(', ')][self/][/for] })
      Public listInstrument As New List(of Object)({ [for
(aSegment.instrument.tag) separator(', ')][self/][/for] })
      Public fluid As New [aSegment.fluid.name/]
```

```
      Public Sub New()

      End Sub
      Public Sub New(Byval name as String,
                          Byval diameter as Double,
                  Byval thickness as Double,
                          Byval firstKm as Double,
                          Byval lastKm as Double,
                  Byval listStation as List(of Object),
                          Byval listInstrument as List(of Object),
                          Byval fluid as Object)

            Me.name = name
            Me.diameter = diameter
            Me.thickness = thickness
            Me.firstKm = firstKm
            Me.lastKm = lastKm
            Me.listStation = listStation
            Me.listInstrument = listInstrument
            Me.fluid = fluid
      End Sub
End Class
[/file]
[/template]
```

# Geração do ElevationProfile e do Fluid

```
[template public
generateElevationProfile(aElevationProfile:ElevationProfile) post (trim())]

[file (aElevationProfile.name + '.vb', false, 'UTF-8')]

Public Class [aElevationProfile.name/]

      Public name as String = "[aElevationProfile.name/]"
      Public initialElevation as Double =
[aElevationProfile.initialElevation/]
      Public finalElevation as Double = [aElevationProfile.finalElevation/]
      Public firstKm as Double = [aElevationProfile.initialKm/]
      Public lastKm as Double = [aElevationProfile.finalKm/]
      Public invertProfile as Boolean = [aElevationProfile.invertProfile/]

      Public Sub New()

      End Sub

      Public Sub New(Byval name as String,
                         Byval initialElevation as Double,
                         Byval finalElevation as Double,
                         Byval firstKm as Double,
                         Byval lastKm as Double,
                         Byval invertProfile as Boolean
                     )

            Me.name = name
            Me.initialElevation = initialElevation
            Me.finalElevation = finalElevation
            Me.firstKm = firstKm
            Me.lastKm = lastKm
            Me.invertProfile = invertProfile

      End Sub

End Class

[/file]
[/template]
```

```
[template public generateFluids(aFluid:Fluid) post (trim())]

[file (aFluid.name + '.vb', false, 'UTF-8')]

Public Class [aFluid.name/]

      Public name as String = "[aFluid.name/]"
      Public density as Double = [aFluid.density/]
      Public id as String = "[aFluid.id/]"
      Public kmInterface as Double = [aFluid.kmInterface/]
      Public vaporPressure as Double = [aFluid.vaporPressure/]
      Public viscosity as Double = [aFluid.viscosity/]
      Public volume as Double = [aFluid.volume/]

      Public Sub New()

      End Sub

      Public Sub New(Byval name as String,
                         Byval id as String,
                         Byval viscosity as Double,
                         Byval density as Double,
                         Byval kmInterface as Double,
                         Byval vaporPressure as Double,
                         Byval volume as Double)

            Me.name = name
            Me.density = density
            Me.id = id
            Me.kmInterface = kmInterface
            Me.vaporPressure = vaporPressure
            Me.viscosity = viscosity
            Me.volume = volume

      End Sub

End Class
```

# Geração de estações e instrumentos

```
[template public generateStations(aStation:Station) post (trim())]

[file (aStation.name + '.vb', false, 'UTF-8')]

Public Class [aStation.name/]

    Public name as String = "[aStation.name/]"
    Public km as Double = [aStation.km/]
    Public direction as String = "[aStation.direction/]"

    Public Sub New()

    End Sub

    Public Sub New(Byval name as String,
                   Byval km as Double,
                   Byval direction as String)

        Me.name = name
        Me.km = km
        Me.direction = direction

    End Sub

End Class
```

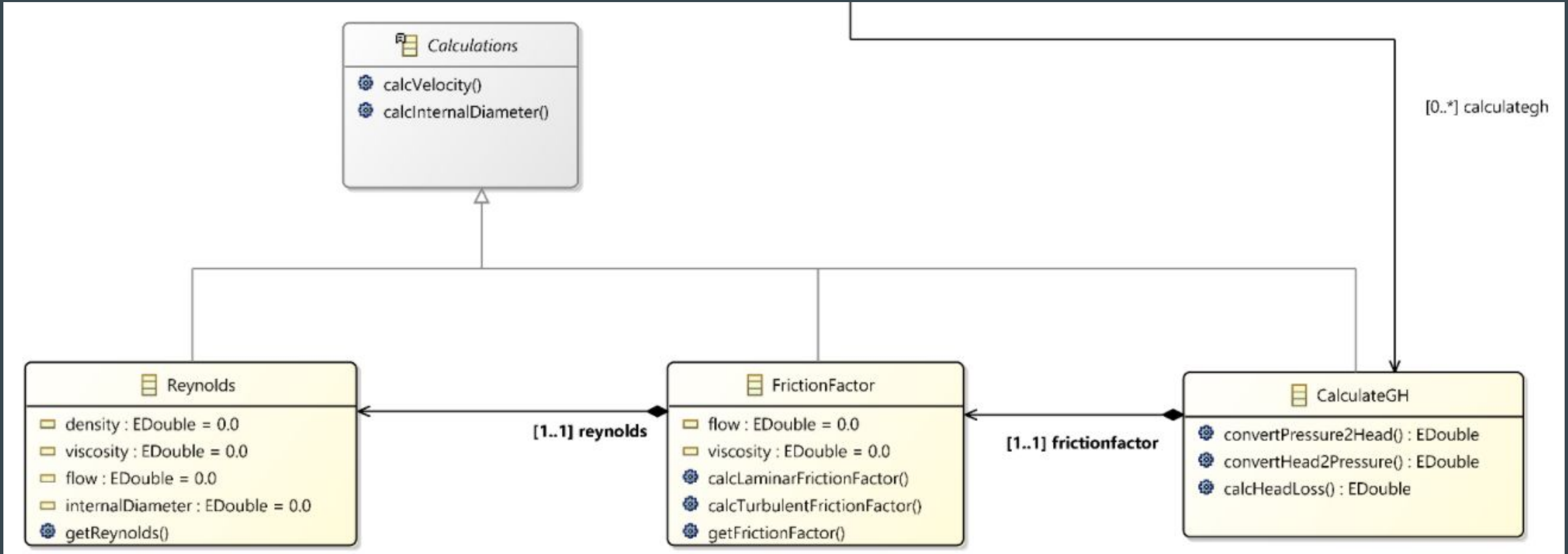
```
[template public generateInstruments(aInstrument:Instrument) post (trim())]

[file (aInstrument.tag + '.vb', false, 'UTF-8')]

Public Class [aInstrument.tag/]

    Public name as String = "[aInstrument.tag/]"
    Public km as Double = [aInstrument.km/]
    Public type as String = "[aInstrument.type/]"
    Public value as Double = [aInstrument.value/]

    Public Sub New()
    End Sub

    Public Sub New(Byval name as String,
                   Byval km as Double,
                   Byval type as String,
                   Byval value as Double)

        Me.name = name
        Me.km = km
        Me.type = type
        Me.value = value

    End Sub
End Class
[/file]
[/template]
```

# InicializaGeral - inicialmente criada para instanciar tudo

```
[template public generateInicializaGeral(pipe : PipelineSystem) ]
[comment TODO Auto-generated template stub/]
[file ('InicializaGeral.vb' , false, 'UTF-8')]
Public Class InicializaGeral

	Public [pipe.name/] As New [pipe.name/]
[comment]
	[for (p : Pipeline | pipe.pipeline)]
	Public [p.name/] As New [p.name/]
	[/for]

[for (p : Pipeline | pipe.pipeline)]
	Public [p.name/] As New [p.name/]
	[for (s : Segment | p.segment)]
	Public [s.name/] As New [s.name/]
		[for (st : Station | s.station)]
	Public [st.name/] As New [st.name/]
		[/for]
		[for (inst : Instrument | s.instrument)]
	Public [inst.tag/] As New [inst.tag/]
		[/for]
		[for (f : Fluid | s.fluid)]
	Public [f.name/] As New [f.name/]
		[/for]
	[/for]
	[for (e : ElevationProfile | p.elevationprofile)]
	Public [e.name/] As New [e.name/]
	[/for]
[/for][/comment]

	Public Sub New()

	End Sub


'[protected (pipe.name)]
'Main code
'[/protected]

End Class
[/file]
[/template]
```

```
[template public generateInicializaGeral(pipe : PipelineSystem) ]
[comment TODO Auto-generated template stub/]
[file ('InicializaGeral.vb' , false, 'UTF-8')]
Public Class InicializaGeral

	Public [pipe.name/] As New [pipe.name/]

	Public Sub New()

	End Sub


'[protected (pipe.name)]
'Main code
'[/protected]

End Class
[/file]
[/template]
```

# Parte do cálculo



# Classe “estática” Reynolds

```
[template public generateReynolds(p : String) ]
[file (p + '.vb', false, 'UTF-8')]

Public Class Reynolds
      Inherits Calculations

      Public visc as Double
      Public velocity as Double
      Public internalDiameter as Double

      Public Sub New()

      End Sub

      Public Sub New(ByVal visc As Double,
                     ByVal flow As Double,
                     ByVal externalDiameter As Double,
                    ByVal thickness As Double)

        Me.visc = visc
        Me.velocity = calcVelocity(externalDiameter, thickness, flow)
        Me.internalDiameter =
UnitConversion.ConvertInches2Meter(calcInternalDiameter(externalDiameter,
thickness))

    End Sub
```

```
      '' <summary>
    ''' Calculates adimensional Reynolds Number
    ''' </summary>
    ''' <param name="velocity">must be in SI (m/s)</param>
    ''' <param name="internalDiameter">must be in SI (m)</param>
    ''' <returns>Reynolds Number</returns>
    Public Function calcReynolds(ByVal velocity As Double,
                                 ByVal internalDiameter As Double)

        Return velocity * internalDiameter / (visc * 0.000001)

    End Function

     Public Function getReynolds()

        Return velocity * internalDiameter / (visc * 0.000001)

    End Function
End Class
[/file]
[/template]
```

# Teste final do modelo em Vb.Net

- Solução 'PipeSim' (1 projeto)
  - PipeSim
    - My Project
    - Referências
    - App.config
    - CalculateGH.vb
    - Calculations.vb
    - FrictionFactor.vb
    - FT001.vb
    - GLNA.vb
    - InicializaGeral.vb
    - Module1.vb
    - OSBRA.vb
    - PIR.vb
    - PLN.vb
    - PLN2PIR.vb
    - PLN2PIRp.vb
    - PLN2PIRp_0.vb
    - PLN2PIRp_1.vb
    - PT001.vb
    - PT002.vb
    - Reynolds.vb
    - S500.vb
    - UnitConversion.vb
    - XV.vb

```
Module Module1

    Sub Main()

        Dim init As New InicializaGeral

        Dim GH1 As New CalculateGH()

        GH1.CalculateAllGH(init.OSBRA)

        Console.ReadKey()

    End Sub

End Module
```

# Caso 1: massas específicas iguais e vazão de 900 $m^3/h$

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PipeLang:PipelineSystem xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:PipeLang="http://www.example.org/PipeLang" name="OSBRA">
  <pipeline name="PLN2PIRp" rugosity="0.045" elasticity="1899731.0"
poisson="0.3">
    &lt;segment diameter="20.0" thickness="0.25" name="PLN2PIRs"/>
    <elevationprofile initialKm="0.0" initialElevation="700.0"
name="PLN2PIR" finalKm="99.0" finalElevation="854.0"/>
    <instrument tag="PT001" type="Pressure" km="0.0">
      <value>72.0</value>
    </instrument>
    <instrument tag="FT001" type="Flow" km="0.0">
      <value>900.0</value>
    </instrument>
    <instrument tag="PT002" type="Pressure" km="99.0">
      <value>7.0</value>
    </instrument>
    <station km="0.0" name="PLN" direction="Sending"/>
    <station km="45.0" name="XV" direction="Sending"/>
    <station km="95.0" name="PIR" direction="Receiving"/>
    <fluid name="S500" density="840.0" volume="10000.0" id="2702200001"
viscosity="345.0" vaporPressure="0.0"/>
    <fluid name="GLNA" density="840.0" volume="-1.0" id="2702200001"
viscosity="1.0" vaporPressure="0.0"/>
    <calculategh/>
  </pipeline>
</PipeLang:PipelineSystem>
```

C:\Users\thiag\source\repos\PipeSim\PipeSim\bin\Debug\PipeSim.exe

```
Sistema OSBRA
Duto PLN2PIRp
Valor do head inicial = 1557
Valor da pressao inicial = 72
Numero de Reynolds para o segmento PLN2PIRp_0 = 1863
Fator de atrito para o segmento PLN2PIRp_0 = 0,0457
Head inicial = 1557
Head final = 1145
Densidade = 0,84
Numero de Reynolds para o segmento PLN2PIRp_1 = 642661
Fator de atrito para o segmento PLN2PIRp_1 = 0,0125
Head inicial = 1145
Head final = 1043
Densidade = 0,84
Press?o final = 15,92
```

# Caso 2: massas específicas iguais e vazão de $0 m^3/h$

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PipeLang:PipelineSystem xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:PipeLang="http://www.example.org/PipeLang" name="OSBRA">
  <pipeline name="PLN2PIRp" rugosity="0.045" elasticity="1899731.0"
poisson="0.3">
    &lt;segment diameter="20.0" thickness="0.25" name="PLN2PIRs"/>
    <elevationprofile initialKm="0.0" initialElevation="700.0"
name="PLN2PIR" finalKm="99.0" finalElevation="854.0"/>
    <instrument tag="PT001" type="Pressure" km="0.0">
      <value>72.0</value>
    </instrument>
    <instrument tag="FT001" type="Flow" km="0.0">
      <value>0.0</value>
    </instrument>
    <instrument tag="PT002" type="Pressure" km="99.0">
      <value>7.0</value>
    </instrument>
    <station km="0.0" name="PLN" direction="Sending"/>
    <station km="45.0" name="XV" direction="Sending"/>
    <station km="95.0" name="PIR" direction="Receiving"/>
    <fluid name="S500" density="840.0" volume="10000.0" id="2702200001"
viscosity="345.0" vaporPressure="0.0"/>
    <fluid name="GLNA" density="840.0" volume="-1.0" id="2702200001"
viscosity="1.0" vaporPressure="0.0"/>
    <calculategh/>
  </pipeline>
</PipeLang:PipelineSystem>
```

C:\Users\thiag\source\repos\PipeSim\PipeSim\bin\Debug\PipeSim.exe

```
Sistema OSBRA
Duto PLN2PIRp
Valor do head inicial = 1557
Valor da pressao inicial = 72
Numero de Reynolds para o segmento PLN2PIRp_0 = 0
Fator de atrito para o segmento PLN2PIRp_0 = 0
Head inicial = 1557
Head final = 1557
Densidade = 0,84
Numero de Reynolds para o segmento PLN2PIRp_1 = 0
Fator de atrito para o segmento PLN2PIRp_1 = 0
Head inicial = 1557
Head final = 1557
Densidade = 0,84
Press?o final = 59,06
```

# Caso 3: massas específicas diferentes e vazão de $0m^3/h$

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PipeLang:PipelineSystem xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:PipeLang="http://www.example.org/PipeLang" name="OSBRA">
  <pipeline name="PLN2PIRp" rugosity="0.045" elasticity="1899731.0"
poisson="0.3">
    &lt;segment diameter="20.0" thickness="0.25" name="PLN2PIRs"/>
    <elevationprofile initialKm="0.0" initialElevation="700.0"
name="PLN2PIR" finalKm="99.0" finalElevation="854.0"/>
    <instrument tag="PT001" type="Pressure" km="0.0">
      <value>72.0</value>
    </instrument>
    <instrument tag="FT001" type="Flow" km="0.0">
      <value>0.0</value>
    </instrument>
    <instrument tag="PT002" type="Pressure" km="99.0">
      <value>7.0</value>
    </instrument>
    <station km="0.0" name="PLN" direction="Sending"/>
    <station km="45.0" name="XV" direction="Sending"/>
    <station km="95.0" name="PIR" direction="Receiving"/>
    <fluid name="S500" density="840.0" volume="10000.0" id="2702200001"
viscosity="345.0" vaporPressure="0.0"/>
    <fluid name="GLNA" density="720.0" volume="-1.0" id="2702200001"
viscosity="1.0" vaporPressure="0.0"/>
    <calculategh/>
  </pipeline>
</PipeLang:PipelineSystem>
```

C:\Users\thiag\source\repos\PipeSim\PipeSim\bin\Debug\PipeSim.exe

```
Sistema OSBRA
Duto PLN2PIRp
Valor do head inicial = 1557
Valor da pressao inicial = 72
Numero de Reynolds para o segmento PLN2PIRp_0 = 0
Fator de atrito para o segmento PLN2PIRp_0 = 0
Head inicial = 1557
Head final = 1557
Densidade = 0,84
Numero de Reynolds para o segmento PLN2PIRp_1 = 0
Fator de atrito para o segmento PLN2PIRp_1 = 0
Head inicial = 1817
Head final = 1817
Densidade = 0,72
Press?o final = 69,31
```

# Caso 4: massas específicas diferentes e vazão de 900 $m^3/h$


```xml
<?xml version="1.0" encoding="UTF-8"?>
<PipeLang:PipelineSystem xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:PipeLang="http://www.example.org/PipeLang" name="OSBRA">
  <pipeline name="PLN2PIRp" rugosity="0.045" elasticity="1899731.0"
poisson="0.3">
    &lt;segment diameter="20.0" thickness="0.25" name="PLN2PIRs"/>
    <elevationprofile initialKm="0.0" initialElevation="700.0"
name="PLN2PIR" finalKm="99.0" finalElevation="854.0"/>
    <instrument tag="PT001" type="Pressure" km="0.0">
      <value>72.0</value>
    </instrument>
    <instrument tag="FT001" type="Flow" km="0.0">
      <value>900.0</value>
    </instrument>
    <instrument tag="PT002" type="Pressure" km="99.0">
      <value>7.0</value>
    </instrument>
    <station km="0.0" name="PLN" direction="Sending"/>
    <station km="45.0" name="XV" direction="Sending"/>
    <station km="95.0" name="PIR" direction="Receiving"/>
    <fluid name="S500" density="840.0" volume="10000.0" id="2702200001"
viscosity="345.0" vaporPressure="0.0"/>
    <fluid name="GLNA" density="720.0" volume="-1.0" id="2702200001"
viscosity="1.0" vaporPressure="0.0"/>
    <calculategh/>
  </pipeline>
</PipeLang:PipelineSystem>
```


C:\Users\thiag\source\repos\PipeSim\PipeSim\bin\Debug\PipeSim.exe

```
Sistema OSBRA
Duto PLN2PIRp
Valor do head inicial = 1557
Valor da pressao inicial = 72
Numero de Reynolds para o segmento PLN2PIRp_0 = 1863
Fator de atrito para o segmento PLN2PIRp_0 = 0,0457
Head inicial = 1557
Head final = 1145
Densidade = 0,84
Numero de Reynolds para o segmento PLN2PIRp_1 = 642661
Fator de atrito para o segmento PLN2PIRp_1 = 0,0125
Head inicial = 1145
Head final = 1043
Densidade = 0,84
Press?o final = 15,92
```

# Conclusões e Recomendações de trabalhos futuros

- Muita dificuldade em obter informações específicas sobre o uso das ferramentas na internet
- Problemas na utilização das ferramentas, pouca familiaridade com as mesmas.
- Recomenda-se o uso de controle de versão se possível dentro da própria ferramenta, uma vez que tentativas de mudanças nos metamodelos acarretaram a corrupção dos projetos.
- Como trabalho futuro, outras funcionalidades podem ser adicionadas, tais como os cálculos de correção de vazão para pressão e temperatura, integração da vazão para obtenção do volume, verificação de quebra de coluna, algoritmos de detecção e localização de vazamento. Inserção da temperatura no gradiente hidráulico, etc.

# Conclusões e Recomendações de trabalhos futuros

- Não consegui trabalhar eficientemente com métodos, a geração automática dos mesmos não foi possível.
- Codificação do acceleo deve ser trocada para geração de strings/mensagens em português.
- Não consegui trabalhar com enumerados na transformação M2M.
- Com a geração automática dos arquivos, o código pode ser facilmente testado em um projeto de console.
- Não consegui listar os atributos em uma coleção de forma automática no acceleo, tal como pude observar em exemplos com java, cujo metamodelo era UML ou ECore.