SIMULAÇÃO

"O homem honrado busca justiça. O homem pequeno busca vantagens."
Confúcio. Filósofo chinês
(551 a.C. – 479 a.C.)

Walther Azzolini Júnior
Professor Assistente
Escola de Engenharia de São Carlos
Departamento – Engenharia de Produção
USP – Universidade de São Paulo – Campus São Carlos

EESC – Escola de Engenharia de São Carlos

PLANT SIMULATION – USP

Duplicate & Derive

08/08/2023

wazzolini@sc.usp.br
www.prod.eesc.usp.br

Versão de Hodgson do Algoritmo de Moore – Plant Simulation

Foto: David McGrath

FLUXOGRAMA

AMARRAÇÃO DOS DADOS:

1) cod_Pedido
2) tempo_proc
3) data_entrega

```
is
```

Definir variáveis

```
    linhas : integer;
    i :integer;
do
```

Variável i (número inteiro – número de linhas da tabela)

```
    --importa dados do excel
    tf_entrada.readExcelFile("Pedidos.xlsx");
```

Lendo arquivo *Excel* ⇒ Pedidos.xlsx –

```
    --trata os dados
```

gera dados aleatórios de pedidos

```
    linhas := tf_entrada.YDim - 2;
```

No *Plant Simulation* ⇒ inverte: Y linha e X coluna

```
    for i := 1 to linhas loop
```

– dimensão Y – 2 começa a preencher na linha 2

```
        tf_dados[1,i] := str_to_Num(tf_entrada[1,i+2]); --cod_Pedido
        tf_dados[2,i] := str_to_Num(tf_entrada[2,i+2]); --tempo_proc (min)
        tf_dados[3,i] := str_to_Num(tf_entrada[3,i+2]); --data_entrega
    next;
end;
```

anotações

Para i = 1 para linhas *loop* – percorrer da linha 1 até a última linha preenchida ou com registro dos campos indicados

uso do ponteiro *for*

Tabela dados – colunas 1, 2 e 3 sendo a linha i considerada até a linha em que há registro

Transformar os dados da tabela entrada de *string* para numérico quando necessários. Dados das colunas 1, 2 e 3 a partir da linha 3, ou seja, considerar a linha vazia não considera para registro e a linha de cabeçalho

Os dados aleatórios dos pedidos são gerados no *excel* a partir do procedimento de gerar dados aleatórios do *excel*.

Início gerar dados aleatórios no excel.

Método para importação da planilha Excel Pedidos.xlsx para a tabela tf_entrada com inserção de dados nos campos definidos a partir da 2ª linha

cod_Pedido	tempo_proc	data_entrega
1	27	0
2	16	4

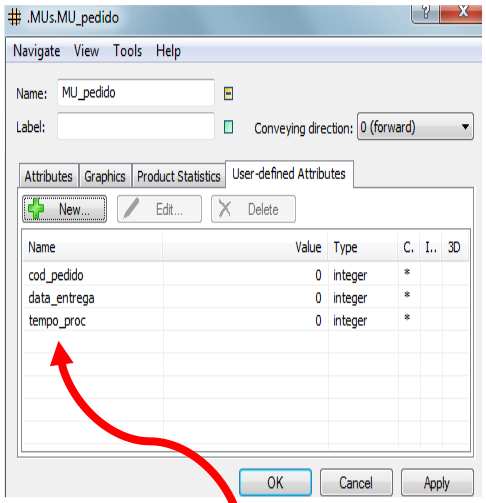
mtd_importacao

```
is
    pedido : object;
    i :integer;
do
    for i:=1 to tf_dados.YDim loop
        pedido := .MUs.mu_pedido.create(Sorter);
        pedido.cod_pedido    := tf_dados[1,i];
        pedido.tempo_proc    := tf_dados[2,i];
        pedido.data_entrega  := tf_dados[3,i];
    next;
    Sorter.sort;
end;
```

mtd_criaPedidos

1) tf_entrada ⇒
   a) cod_Pedido
   b) tempo_proc
   c) data_entrega
2) tf_dados ⇒
   a) cod_Pedido
   b) tempo_proc
   c) data_entrega

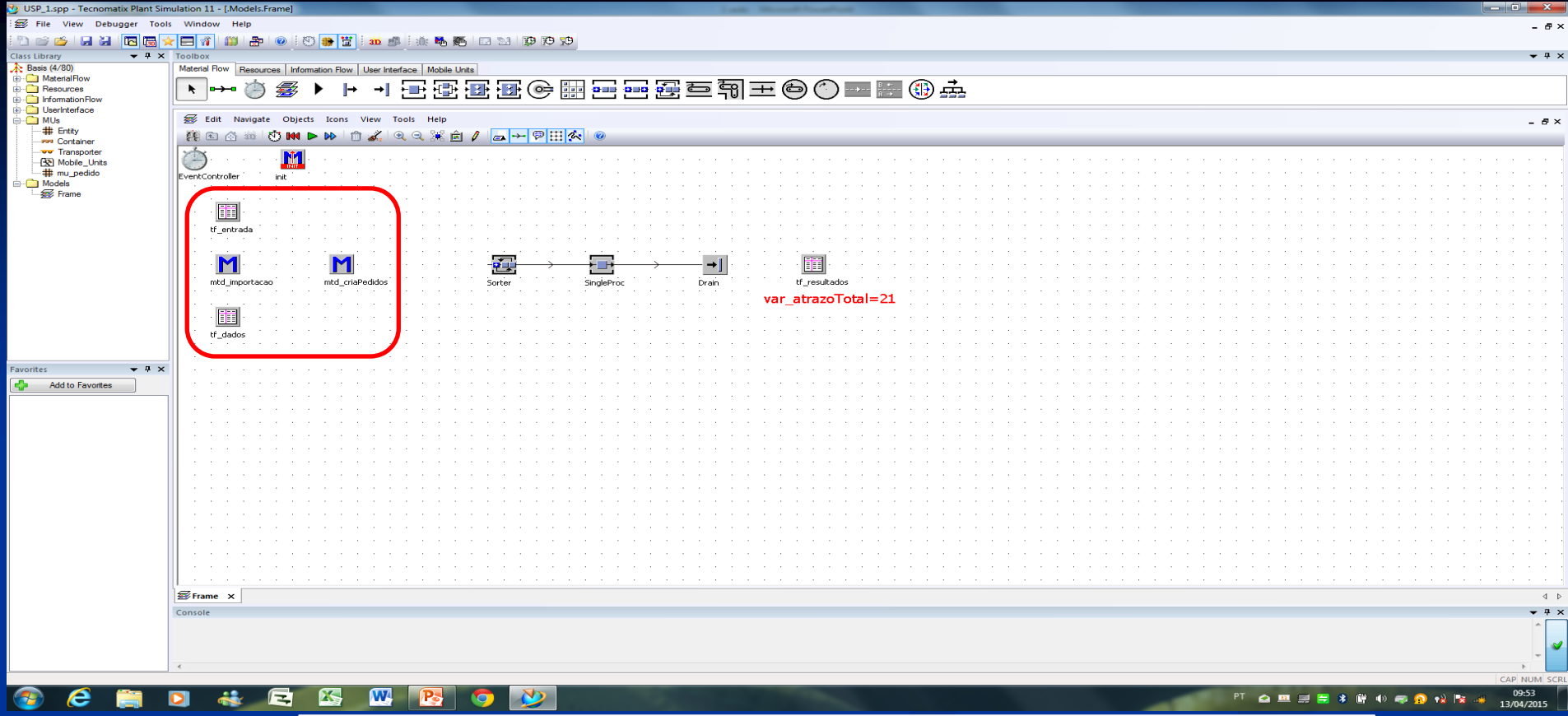
```
is
        linhas : integer;
        i :integer;
do
        --importa dados do excel
        tf_entrada.readExcelFile("Pedidos.xlsx");

        --trataos dados
        linhas := tf_entrada.YDim - 2;

        for i := 1 to linhas loop
                tf_dados[1,i] := str_to_Num(tf_entrada[1,i+2]); --cod_Pedido
                tf_dados[2,i] := str_to_Num(tf_entrada[2,i+2]); --tempo_proc (min)
                tf_dados[3,i] := str_to_Num(tf_entrada[3,i+2]); --data_entrega
        next;
end;

is
        pedido : object;
        i :integer;
do
        for i:=1 to tf_dados.YDim loop
                pedido := .MUs.mu_pedido.create(Sorter);
                pedido.cod_pedido    := tf_dados[1,i];
                pedido.tempo_proc    := tf_dados[2,i];
                pedido.data_entrega  := tf_dados[3,i];
        next;
        Sorter.sort;
end;
```



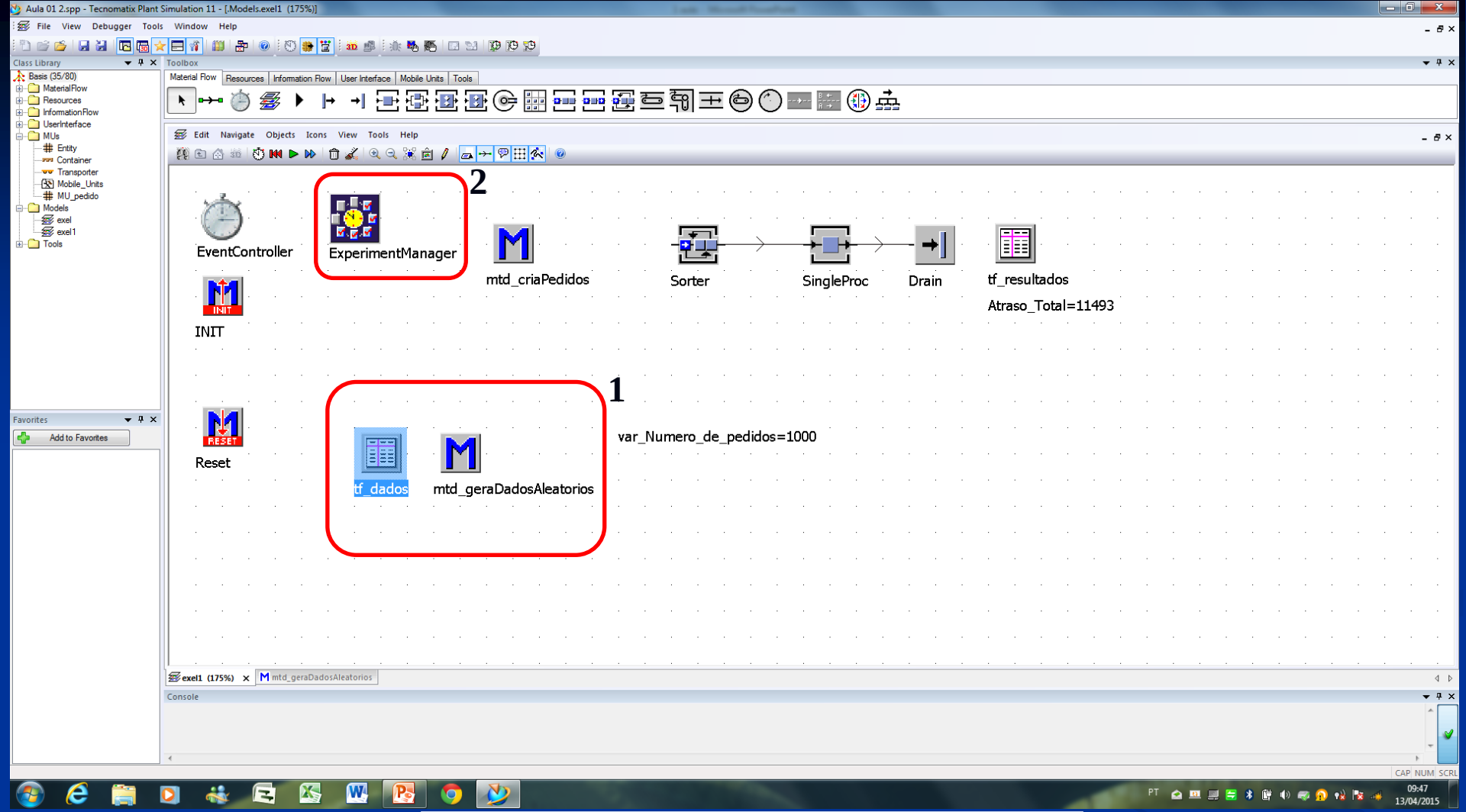
MU_Pedidos

No *Plant Simulation*

MTD_GERADADOSALEATORIOS

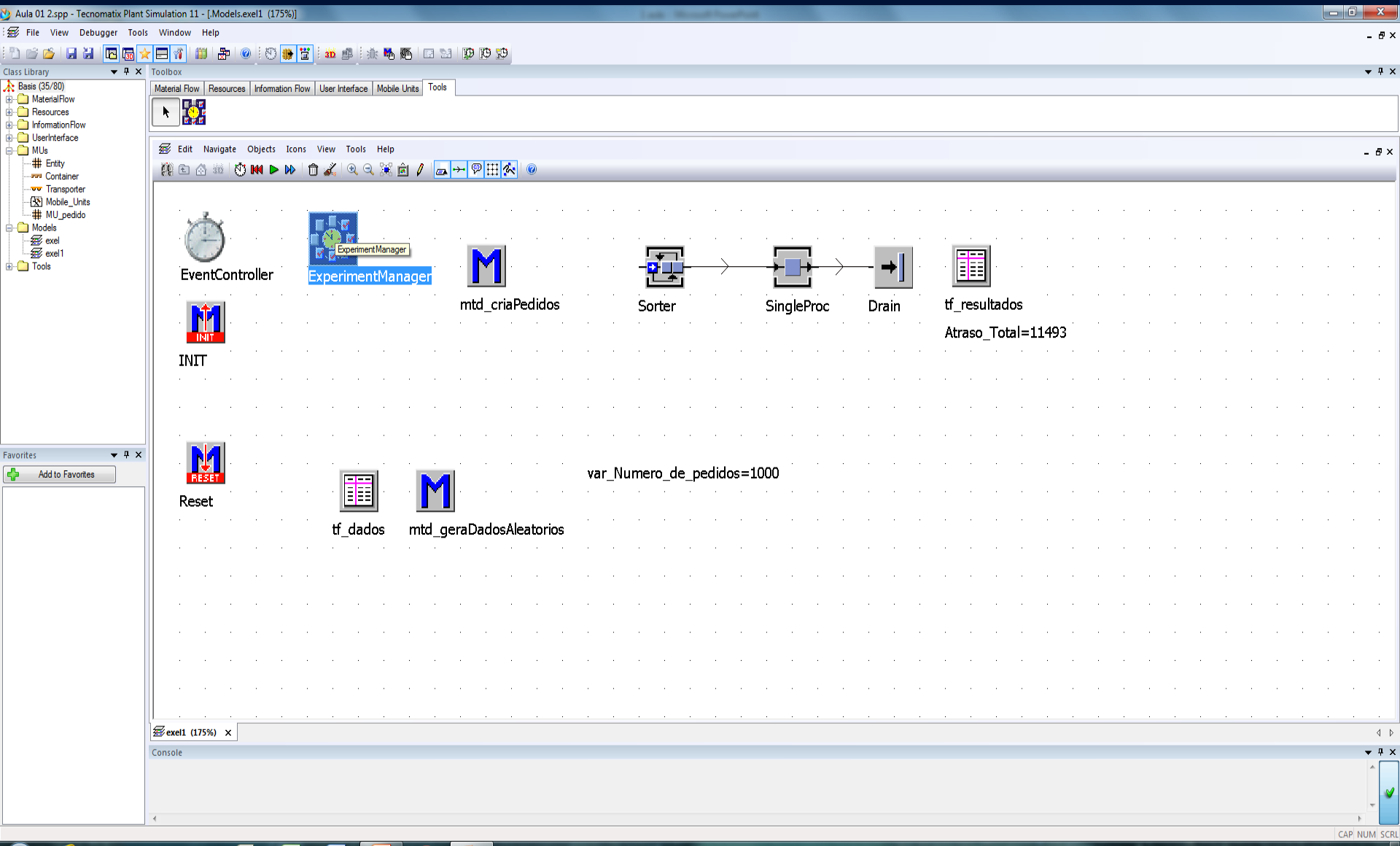


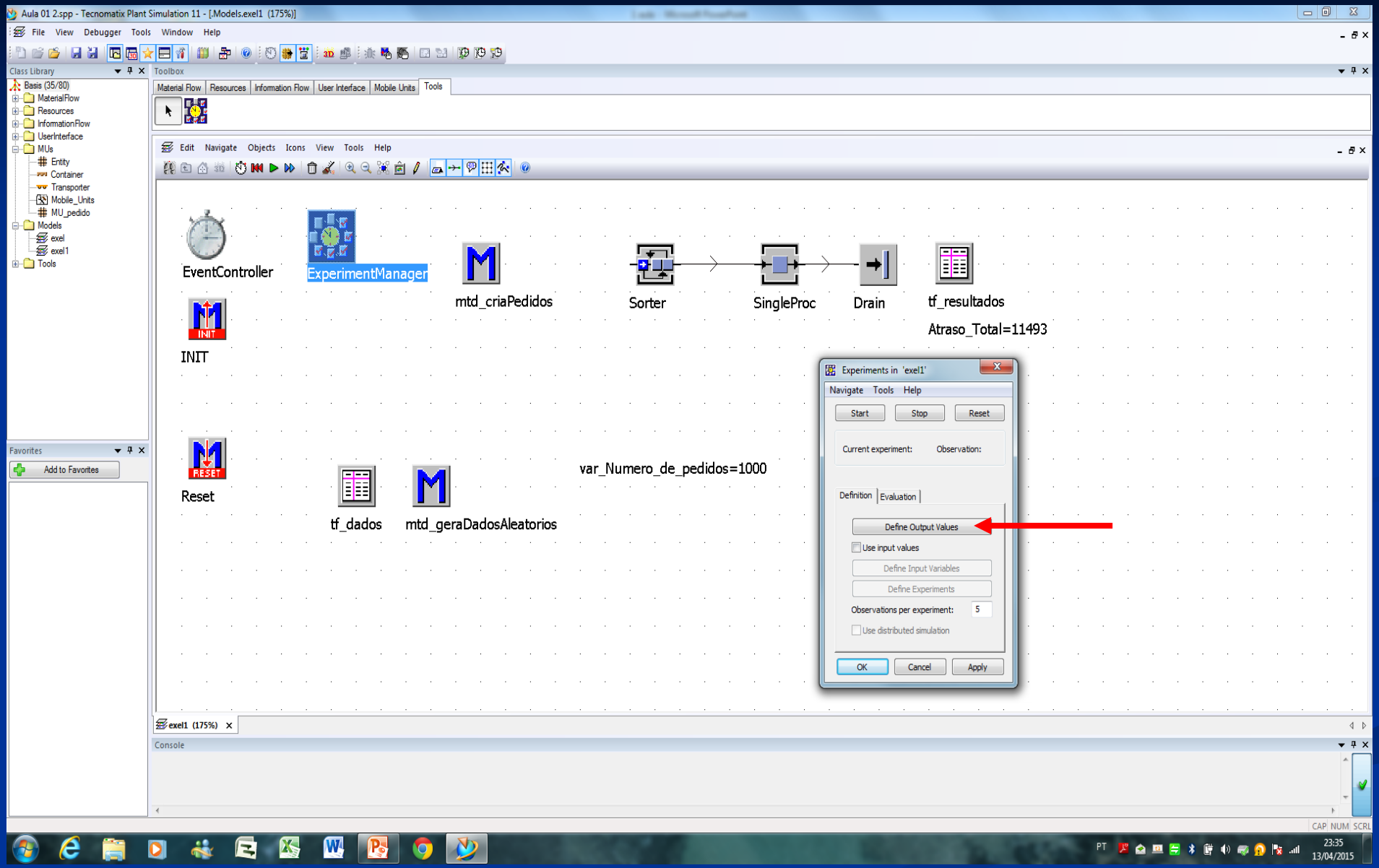

```
is
                linhas : integer;
                i :integer;
do
                --importa dados do excel
                tf_entrada.readExcelFile("Pedidos.xlsx");
                --trata os dados
                linhas := tf_entrada.YDim - 2;
                for i := 1 to linhas loop
                                tf_dados[1,i] := str_to_Num(tf_entrada[1,i+2]); --cod_produto
                                tf_dados[2,i] := str_to_Num(tf_entrada[2,i+2]); --tempo_proc (min)
                                tf_dados[3,i] := str_to_Num(tf_entrada[3,i+2]); --data_entrega
                next;
end;
```

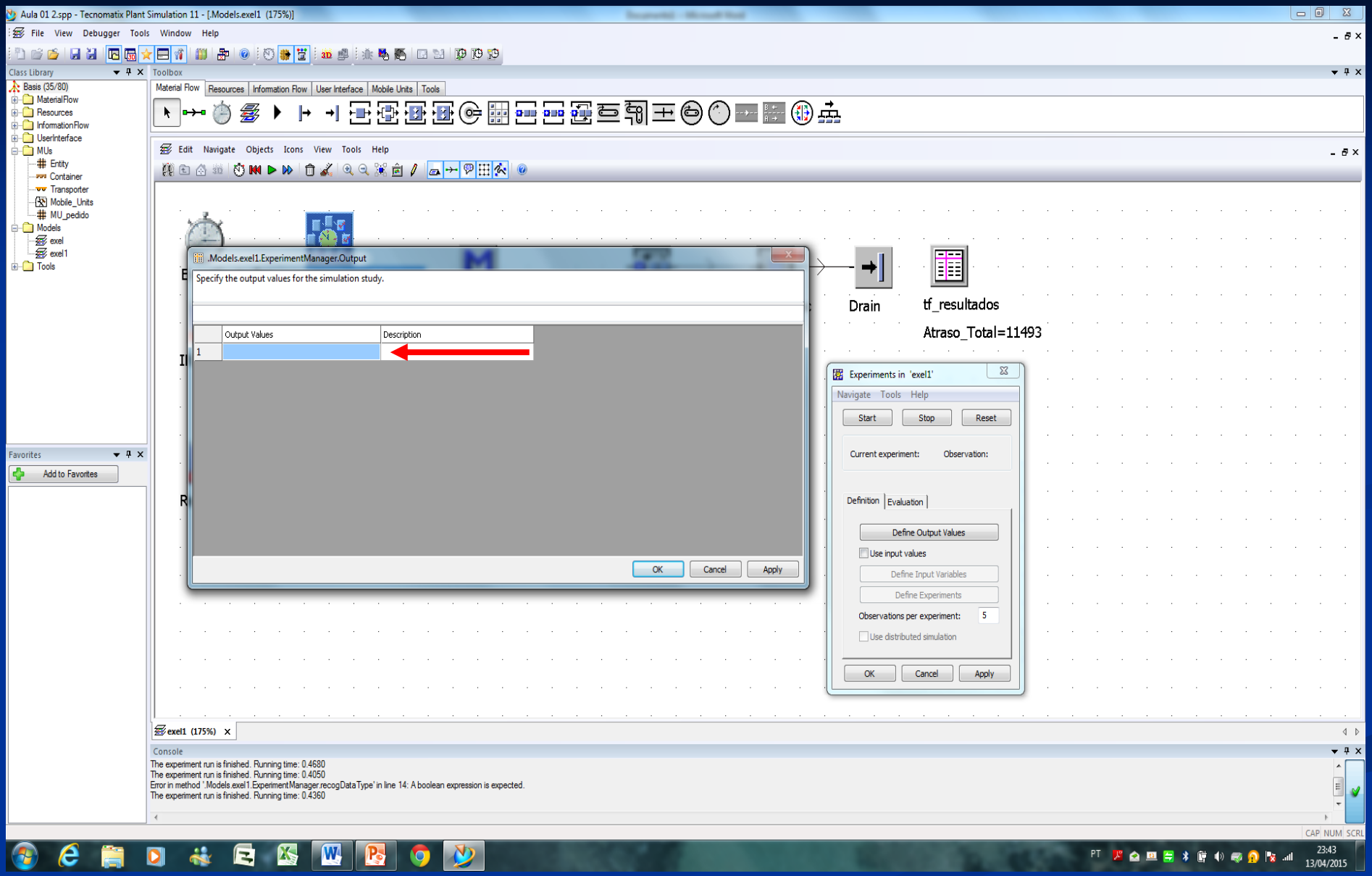


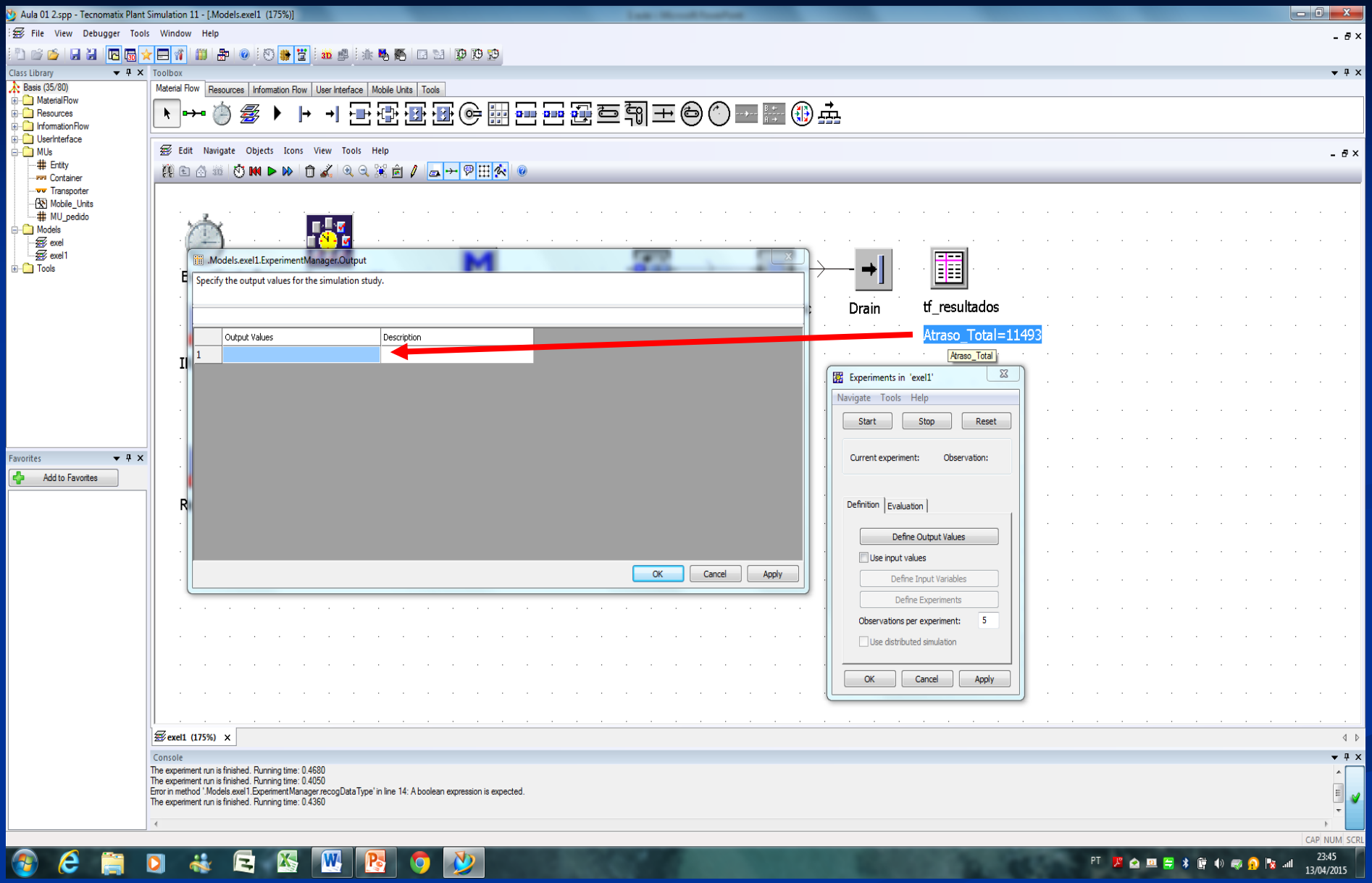
1

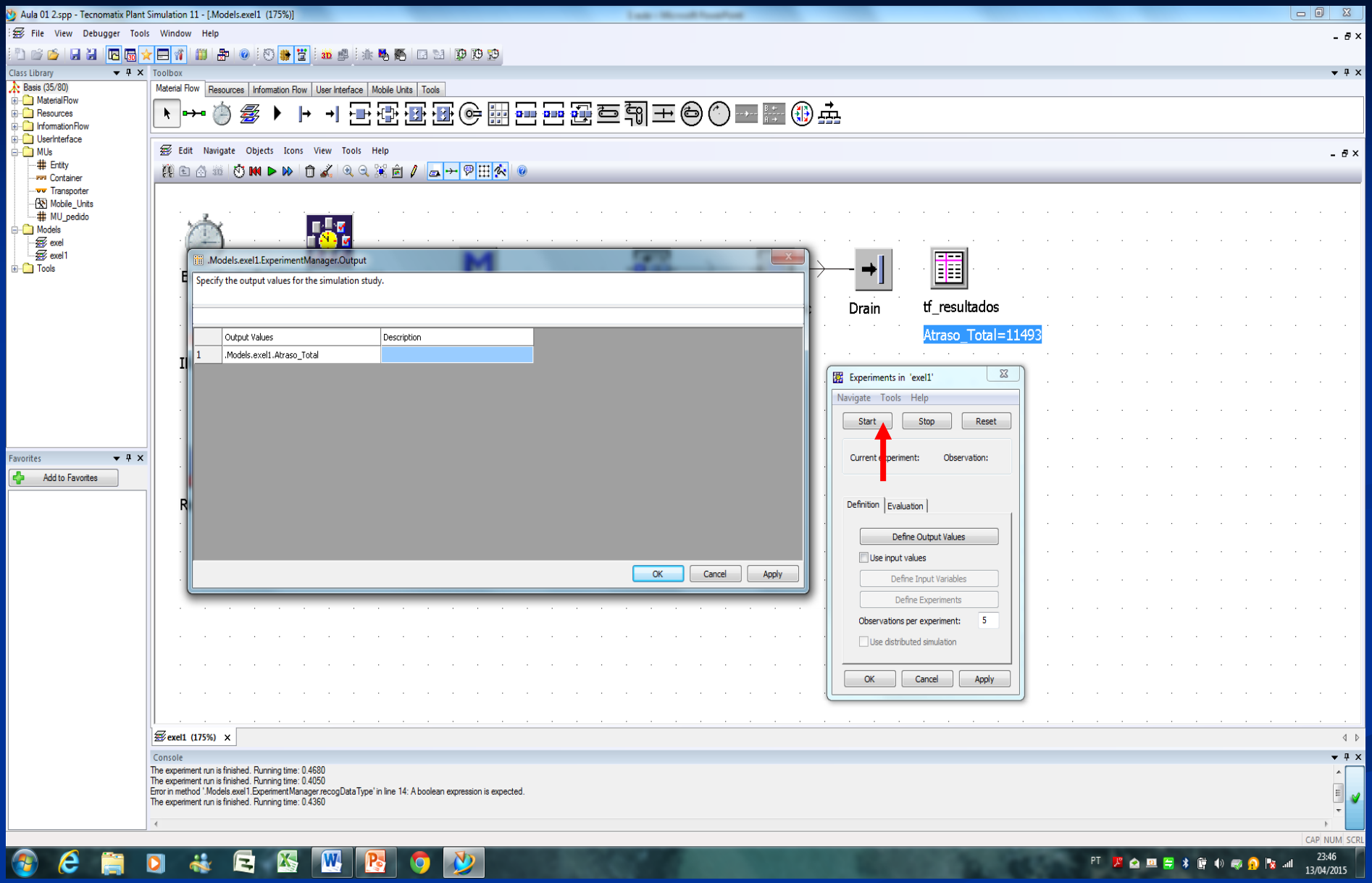
```
is
        i :integer;
do
        for i:=1 to var_Numero_de_pedidos loop
                tf_dados[1,i]:=i;
                tf_dados[2,i]:=z_uniform(1,5,60);
                tf_dados[3,i]:=z_uniform(1,0,6);
        next;
end;
```

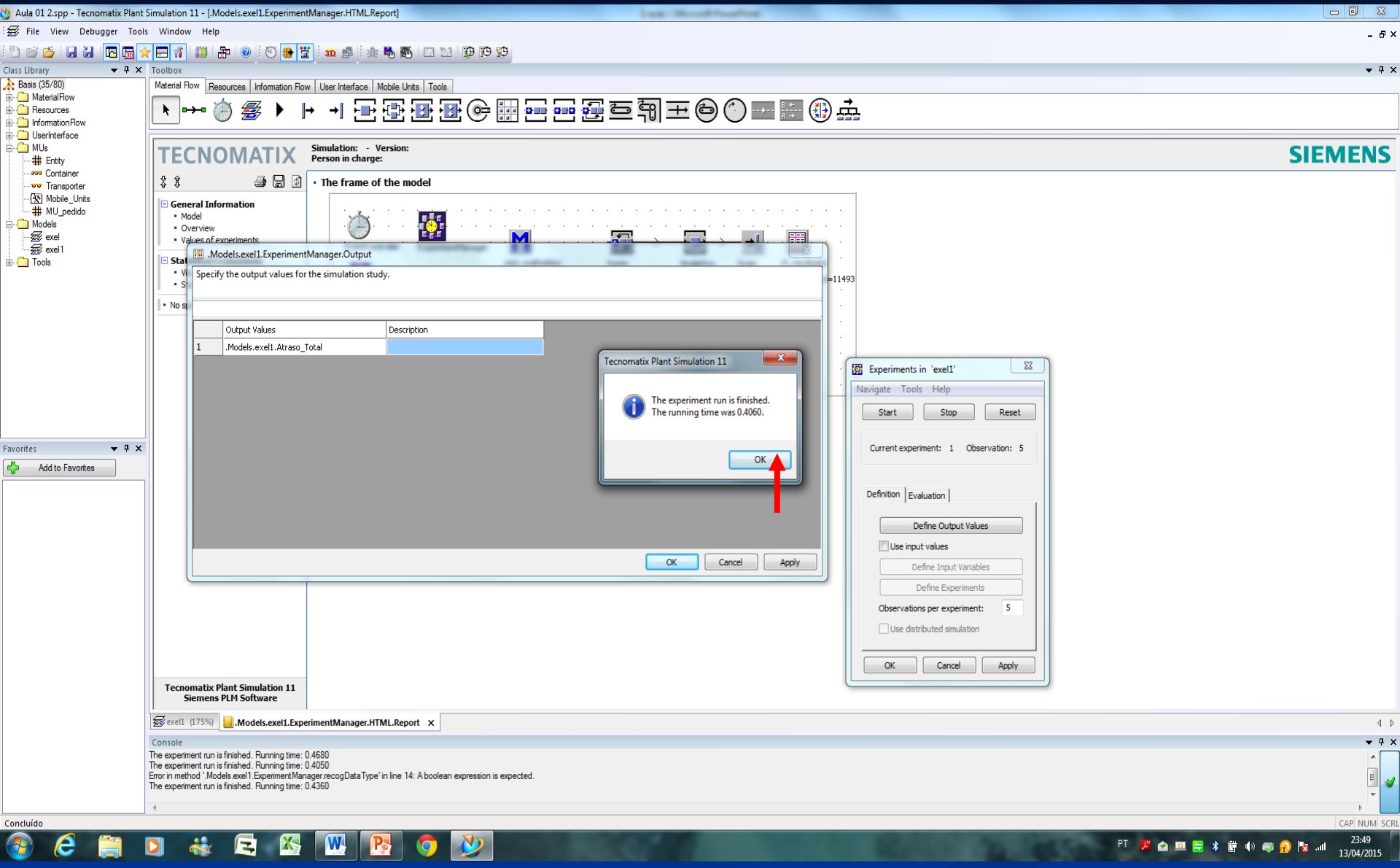

Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1 (175%)]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Edit Navigate Objects Icons View Tools Help
EventController
ExperimentManager
ExperimentManager
mtd_criaPedidos
Sorter
SingleProc
Drain
tf_resultados
Atraso_Total=11493
INIT
INIT
Reset
RESET
tf_dados
mtd_geraDadosAleatorios
var_Numero_de_pedidos=1000
exel1 (175%)
Console
CAP NUM SCRL
PT
23:35
13/04/2015

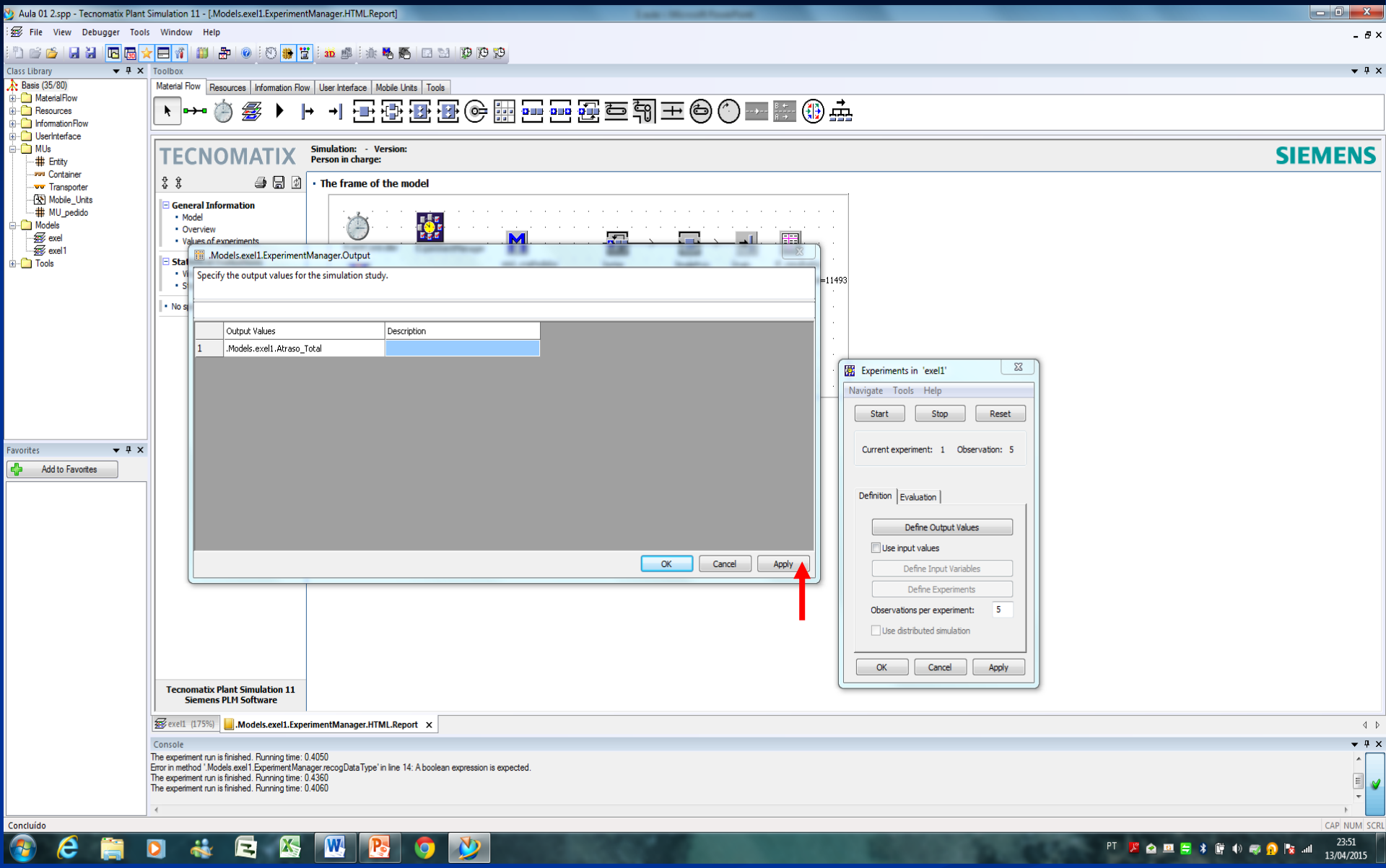

Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1 (175%)]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Edit Navigate Objects Icons View Tools Help
EventController
ExperimentManager
mtd_criaPedidos
Sorter
SingleProc
Drain
tf_resultados
Atraso_Total=11493
INIT
INIT
Reset
RESET
tf_dados
mtd_geraDadosAleatorios
var_Numero_de_pedidos=1000
Experiments in 'exel1'
Navigate Tools Help
Start
Stop
Reset
Current experiment:
Observation:
Definition
Evaluation
Define Output Values
Use input values
Define Input Variables
Define Experiments
Observations per experiment:
5
Use distributed simulation
OK
Cancel
Apply
exel1 (175%)
Console
PT
23:35
13/04/2015

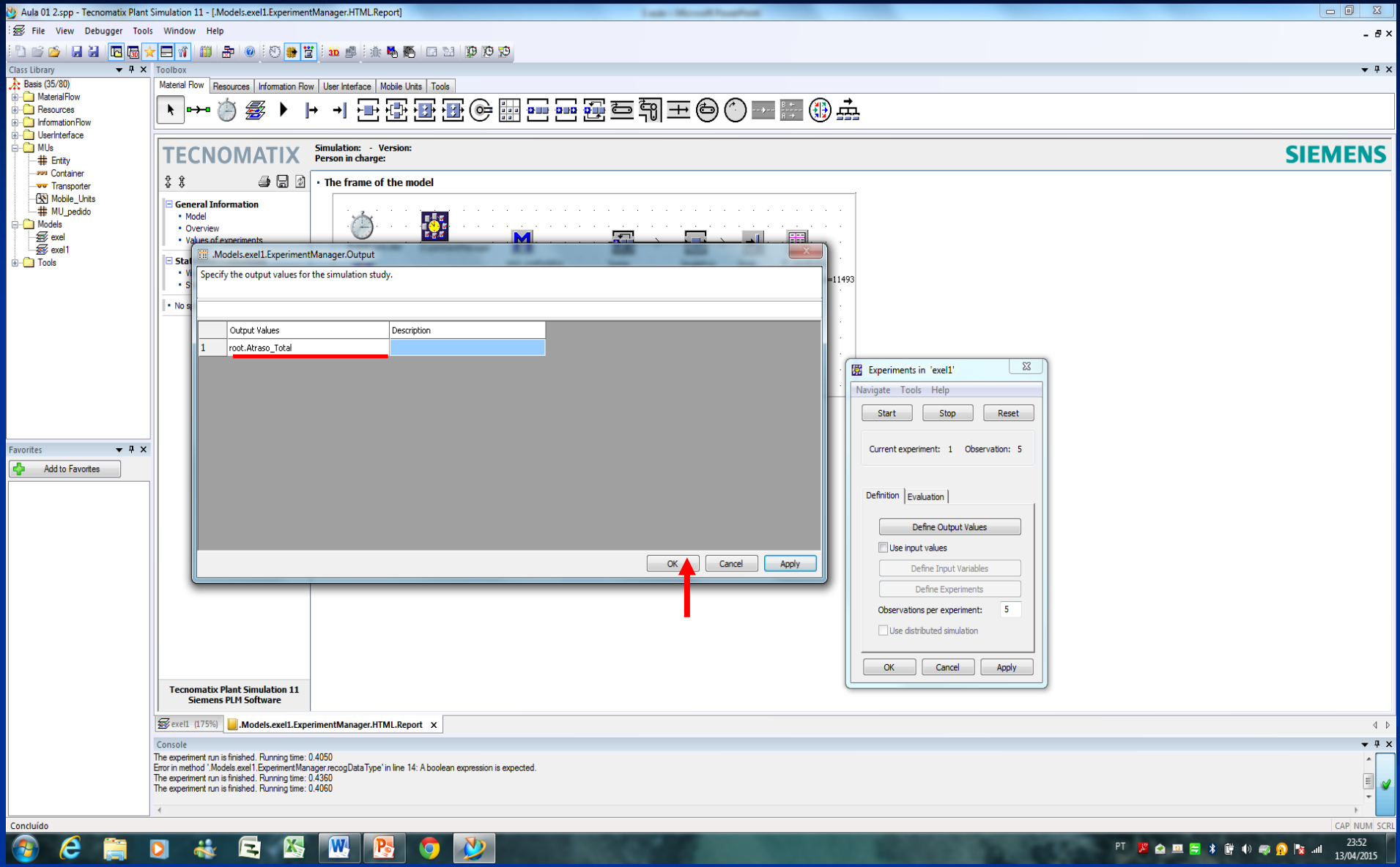

Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1 (175%)]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Edit Navigate Objects Icons View Tools Help
.Models.exel1.ExperimentManager.Output
Specify the output values for the simulation study.
Output Values
Description
1
OK
Cancel
Apply
Drain
tf_resultados
Atraso_Total=11493
Experiments in 'exel1'
Navigate Tools Help
Start
Stop
Reset
Current experiment:
Observation:
Definition
Evaluation
Define Output Values
Use input values
Define Input Variables
Define Experiments
Observations per experiment:
5
Use distributed simulation
OK
Cancel
Apply
exel1 (175%)
Console
The experiment run is finished. Running time: 0.4680
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
CAP NUM SCRL
PT
23:43
13/04/2015

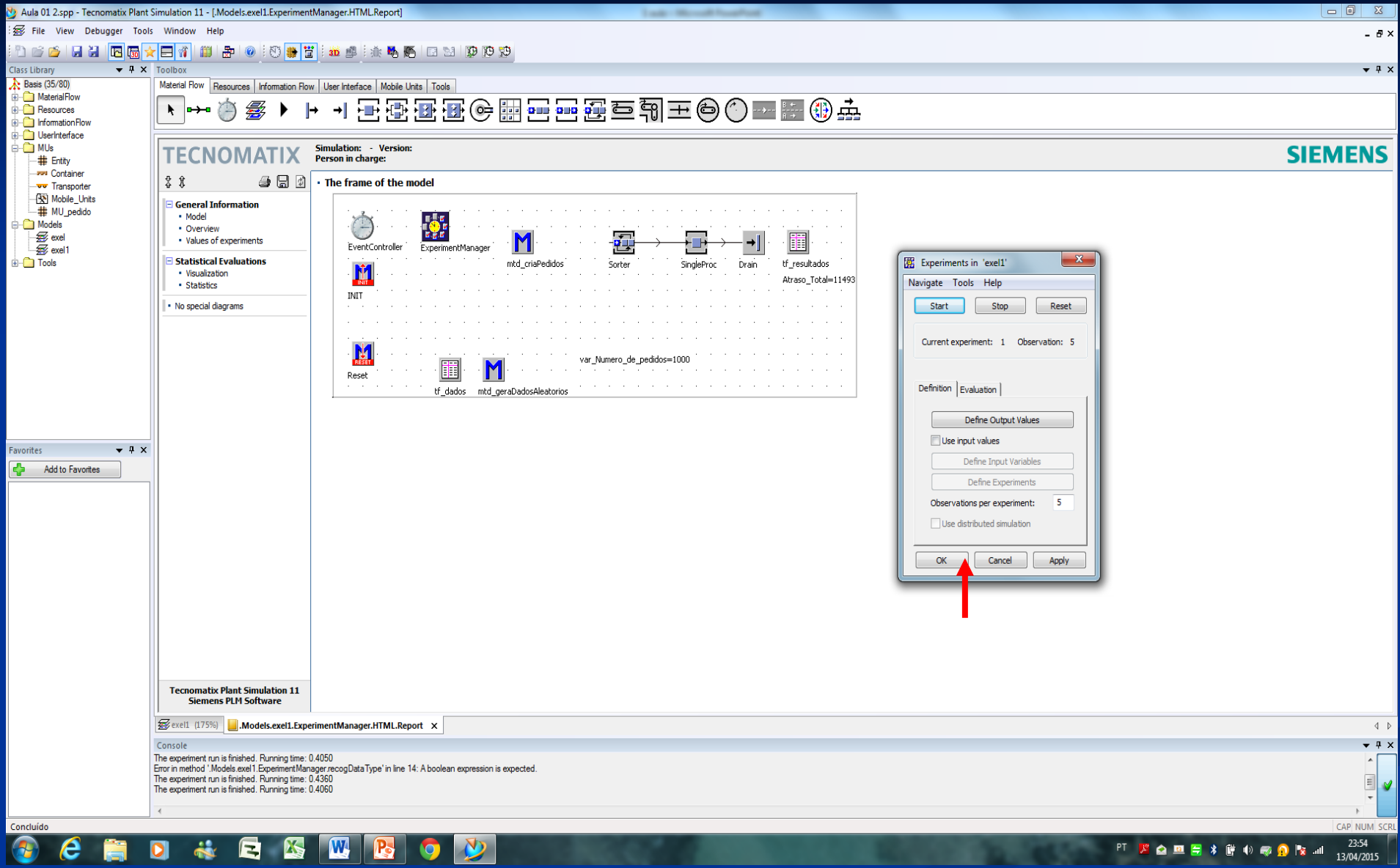

Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1 (175%)]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Edit Navigate Objects Icons View Tools Help
.Models.exel1.ExperimentManager.Output
Specify the output values for the simulation study.
Output Values
Description
1
OK
Cancel
Apply
Drain
tf_resultados
Atraso_Total=11493
Atraso_Total
Experiments in 'exel1'
Navigate Tools Help
Start
Stop
Reset
Current experiment:
Observation:
Definition
Evaluation
Define Output Values
Use input values
Define Input Variables
Define Experiments
Observations per experiment:
5
Use distributed simulation
OK
Cancel
Apply
exel1 (175%)
Console
The experiment run is finished. Running time: 0.4680
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
CAP NUM SCRL
PT
23:45
13/04/2015

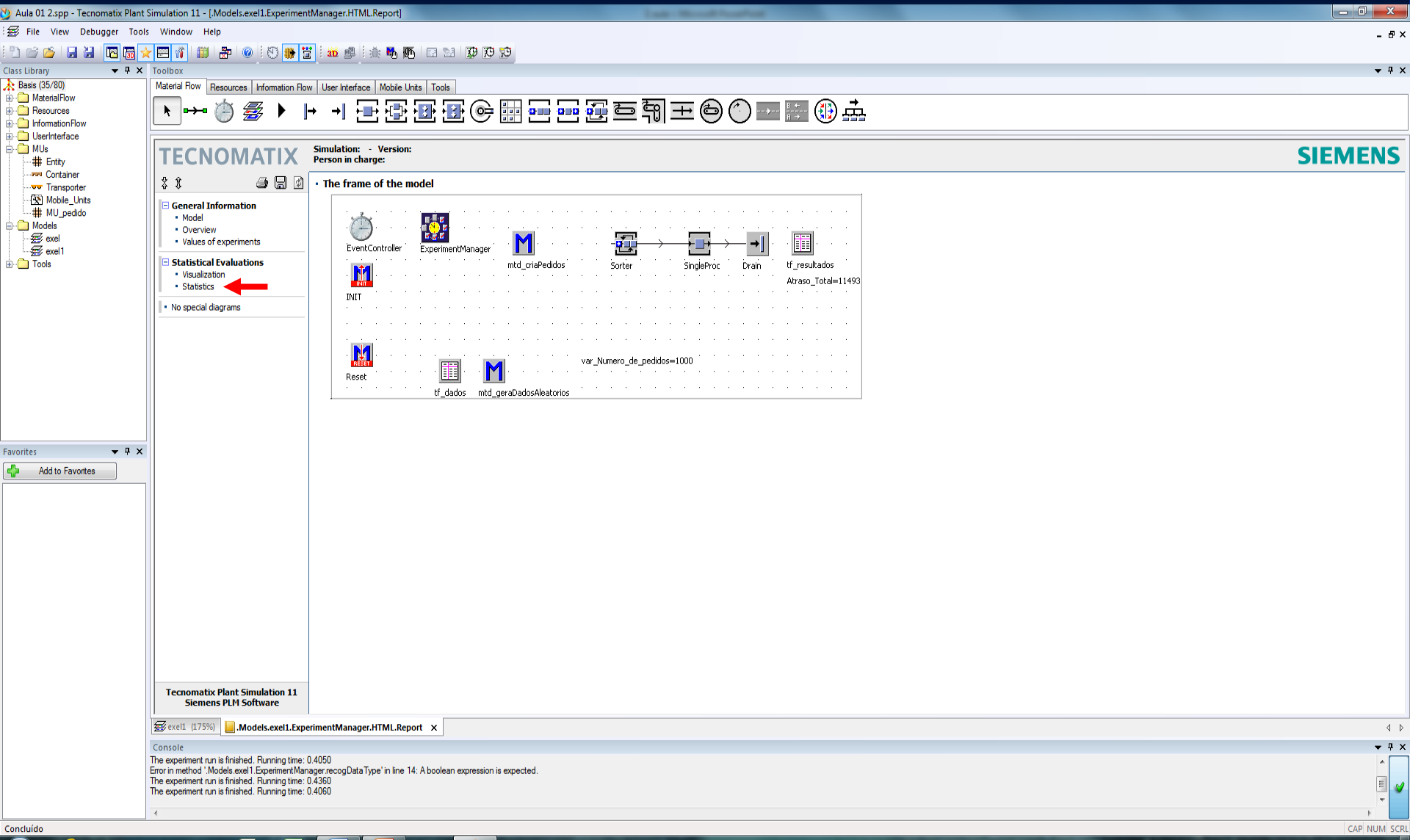

Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1 (175%)]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Edit Navigate Objects Icons View Tools Help
.Models.exel1.ExperimentManager.Output
Specify the output values for the simulation study.
Output Values
Description
1
.Models.exel1.Atraso_Total
OK
Cancel
Apply
Drain
tf_resultados
Atraso_Total=11493
Experiments in 'exel1'
Navigate Tools Help
Start
Stop
Reset
Current experiment:
Observation:
Definition
Evaluation
Define Output Values
Use input values
Define Input Variables
Define Experiments
Observations per experiment:
5
Use distributed simulation
OK
Cancel
Apply
exel1 (175%)
Console
The experiment run is finished. Running time: 0.4680
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
CAP NUM SCRL
PT
23:46
13/04/2015

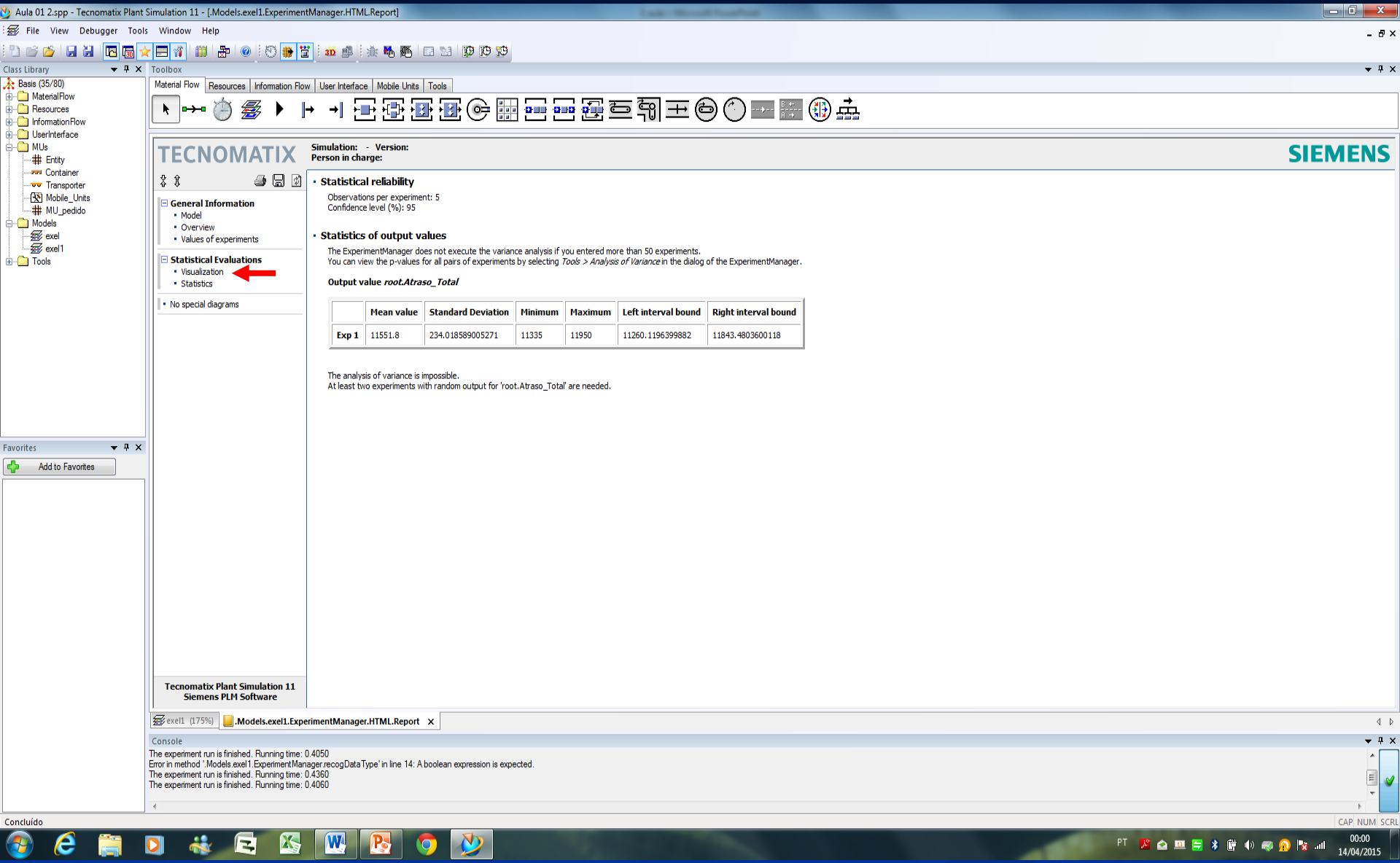

Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1.ExperimentManager.HTML.Report]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InfomationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
TECNOMATIX
Simulation: - Version:
Person in charge:
SIEMENS
The frame of the model
General Information
Model
Overview
Values of experiments
.Models.exel1.ExperimentManager.Output
Specify the output values for the simulation study.
Output Values
Description
1
.Models.exel1.Atraso_Total
OK
Cancel
Apply
Tecnomatix Plant Simulation 11
The experiment run is finished.
The running time was 0.4060.
OK
Experiments in 'exel1'
Navigate Tools Help
Start
Stop
Reset
Current experiment: 1 Observation: 5
Definition
Evaluation
Define Output Values
Use input values
Define Input Variables
Define Experiments
Observations per experiment: 5
Use distributed simulation
OK
Cancel
Apply
Tecnomatix Plant Simulation 11
Siemens PLM Software
exel1 (175%)
.Models.exel1.ExperimentManager.HTML.Report
Console
The experiment run is finished. Running time: 0.4680
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
Concluído
CAP NUM SCRL
PT
23:49
13/04/2015


Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1.ExperimentManager.HTML.Report]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
TECNOMATIX
Simulation: - Version:
Person in charge:
SIEMENS
The frame of the model
General Information
Model
Overview
Values of experiments
.Models.exel1.ExperimentManager.Output
Specify the output values for the simulation study.
Output Values
Description
1
.Models.exel1.Atraso_Total
OK
Cancel
Apply
Experiments in 'exel1'
Navigate Tools Help
Start
Stop
Reset
Current experiment: 1 Observation: 5
Definition
Evaluation
Define Output Values
Use input values
Define Input Variables
Define Experiments
Observations per experiment: 5
Use distributed simulation
OK
Cancel
Apply
Tecnomatix Plant Simulation 11
Siemens PLM Software
exel1 (175%)
.Models.exel1.ExperimentManager.HTML.Report
Console
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
The experiment run is finished. Running time: 0.4060
Concluído
CAP NUM SCRL
PT
23:51
13/04/2015


Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1.ExperimentManager.HTML.Report]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InfomationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
TECNOMATIX
Simulation: - Version:
Person in charge:
SIEMENS
The frame of the model
General Information
Model
Overview
Values of experiments
.Models.exel1.ExperimentManager.Output
Specify the output values for the simulation study.
Output Values
Description
1
root.Atraso_Total
OK
Cancel
Apply
Experiments in 'exel1'
Navigate Tools Help
Start
Stop
Reset
Current experiment: 1 Observation: 5
Definition
Evaluation
Define Output Values
Use input values
Define Input Variables
Define Experiments
Observations per experiment: 5
Use distributed simulation
OK
Cancel
Apply
Tecnomatix Plant Simulation 11
Siemens PLM Software
exel1 (175%)
.Models.exel1.ExperimentManager.HTML.Report
Console
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
The experiment run is finished. Running time: 0.4060
Concluído
CAP NUM SCRL
PT
23:52
13/04/2015


Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1.ExperimentManager.HTML.Report]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InfomationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
TECNOMATIX
Simulation: - Version:
Person in charge:
SIEMENS
The frame of the model
General Information
Model
Overview
Values of experiments
Statistical Evaluations
Visualization
Statistics
No special diagrams
EventController
ExperimentManager
mtd_criaPedidos
Sorter
SingleProc
Drain
tf_resultados
Atraso_Total=11493
INIT
Reset
tf_dados
mtd_geraDadosAleatorios
var_Numero_de_pedidos=1000
Tecnomatix Plant Simulation 11
Siemens PLM Software
exel1 (175%)
.Models.exel1.ExperimentManager.HTML.Report
Console
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
The experiment run is finished. Running time: 0.4060
Concluído
Experiments in 'exel1'
Navigate Tools Help
Start
Stop
Reset
Current experiment: 1 Observation: 5
Definition
Evaluation
Define Output Values
Use input values
Define Input Variables
Define Experiments
Observations per experiment: 5
Use distributed simulation
OK
Cancel
Apply


Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1.ExperimentManager.HTML.Report]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
TECNOMATIX
Simulation: - Version:
Person in charge:
SIEMENS
The frame of the model
General Information
Model
Overview
Values of experiments
Statistical Evaluations
Visualization
Statistics
No special diagrams
EventController
ExperimentManager
mtd_criaPedidos
Sorter
SingleProc
Drain
tf_resultados
Atraso_Total=11493
INIT
Reset
tf_dados
mtd_geraDadosAleatorios
var_Numero_de_pedidos=1000
Tecnomatix Plant Simulation 11
Siemens PLM Software
exel1 (175%)
.Models.exel1.ExperimentManager.HTML.Report
Console
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
The experiment run is finished. Running time: 0.4060
Concluído
CAP NUM SCRL
PT
23:59
13/04/2015


Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1.ExperimentManager.HTML.Report]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
TECNOMATIX
Simulation: - Version:
Person in charge:
SIEMENS
General Information
Model
Overview
Values of experiments
Statistical Evaluations
Visualization
Statistics
No special diagrams
Tecnomatix Plant Simulation 11
Siemens PLM Software
Statistical reliability
Observations per experiment: 5
Confidence level (%): 95
Statistics of output values
The ExperimentManager does not execute the variance analysis if you entered more than 50 experiments.
You can view the p-values for all pairs of experiments by selecting Tools > Analysis of Variance in the dialog of the ExperimentManager.
Output value root.Atraso_Total
Mean value
Standard Deviation
Minimum
Maximum
Left interval bound
Right interval bound
Exp 1
11551.8
234.018589005271
11335
11950
11260.1196399882
11843.4803600118
The analysis of variance is impossible.
At least two experiments with random output for 'root.Atraso_Total' are needed.
exel1 (175%)
.Models.exel1.ExperimentManager.HTML.Report
Console
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
The experiment run is finished. Running time: 0.4060
Concluído
CAP NUM SCRL
PT
00:00
14/04/2015

Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1.ExperimentManager.HTML.Report]

File View Debugger Tools Window Help

Class Library

- Basis (35/80)
  - MaterialFlow
  - Resources
  - InformationFlow
  - UserInterface
  - MUs
    - Entity
    - Container
    - Transporter
    - Mobile_Units
    - MU_pedido
  - Models
    - exel
    - exel1
  - Tools

Favorites

Add to Favorites

Toolbox

Material Flow | Resources | Information Flow | User Interface | Mobile Units | Tools

TECNOMATIX

Simulation: - Version:
Person in charge:

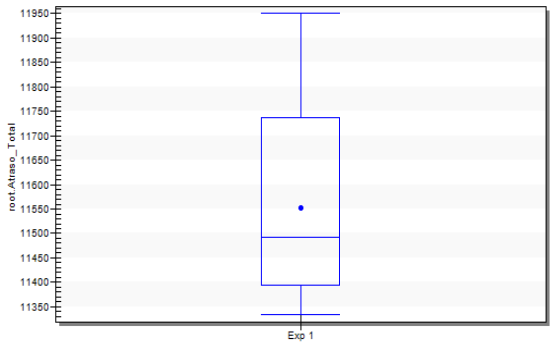
SIEMENS

- **General Information**
  - Model
  - Overview
  - Values of experiments
- **Statistical Evaluations**
  - Visualization
  - Statistics
- No special diagrams

Tecnomatix Plant Simulation 11
Siemens PLM Software

- **Evaluations of the output value *root.Atraso_Tota***

Min-Max intervals with quartiles



exel1 (175%) | .Models.exel1.ExperimentManager.HTML.Report

Console

```
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
The experiment run is finished. Running time: 0.4060
```

Concluído

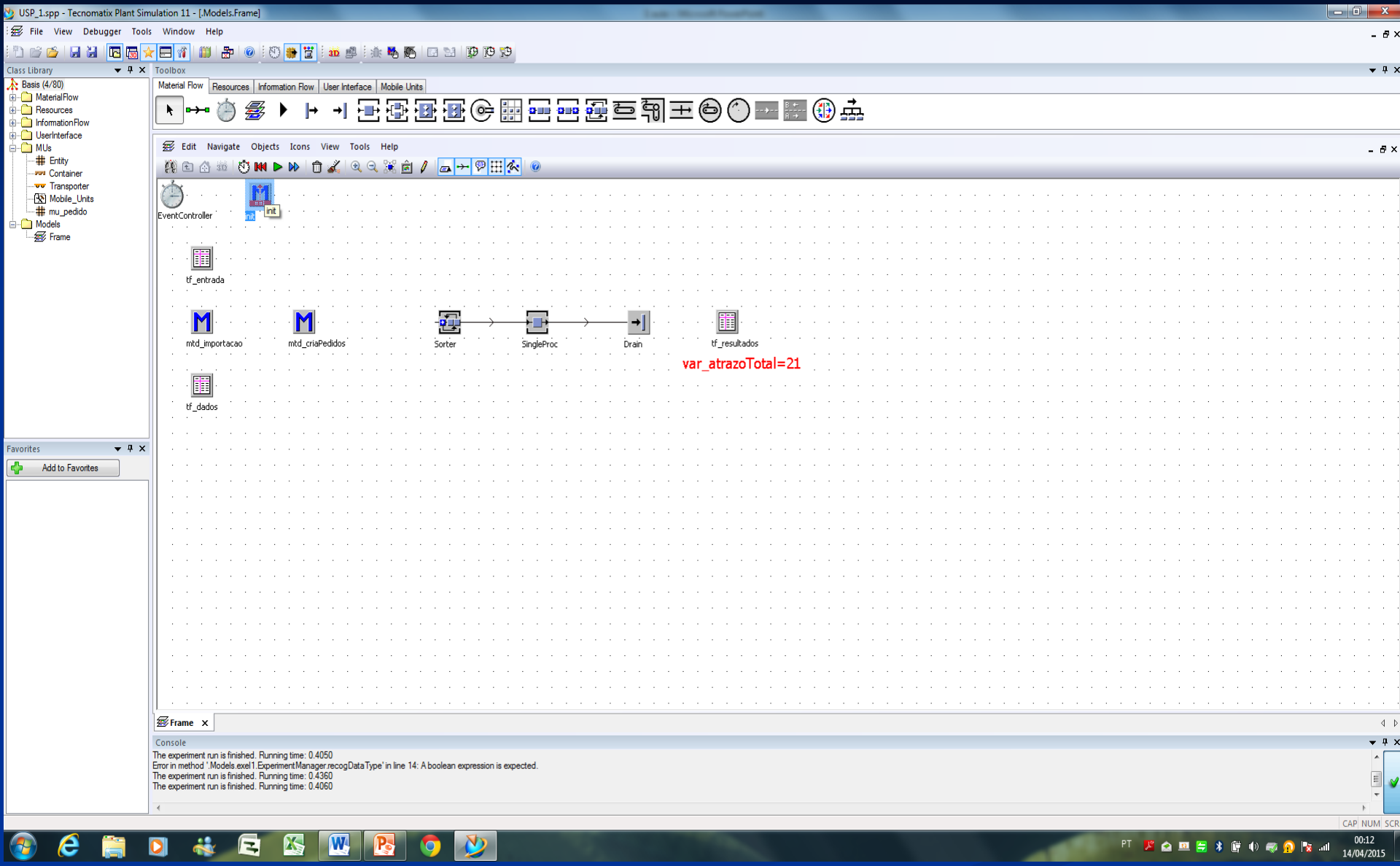
CAP NUM SCRL

PT 00:01 14/04/2015

Métodos

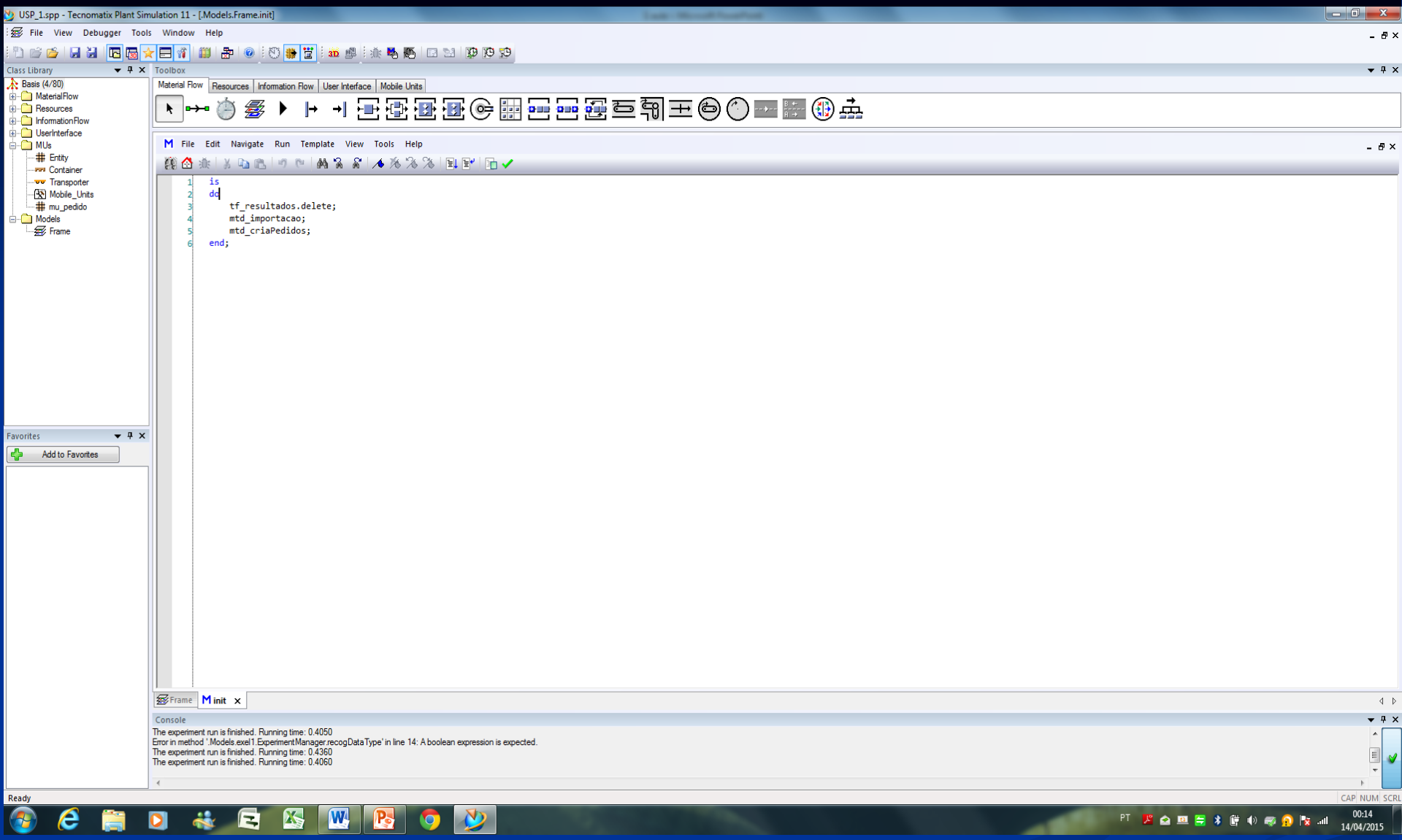
INIT & *Reset*

Basta renomear um método com o nome *INIT* ou *Reset*


USP_1.spp - Tecnomatix Plant Simulation 11 - [.Models.Frame]
File View Debugger Tools Window Help
Class Library
Basis (4/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
mu_pedido
Models
Frame
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Edit Navigate Objects Icons View Tools Help
EventController
init
tf_entrada
mtd_importacao
mtd_criaPedidos
Sorter
SingleProc
Drain
tf_resultados
var_atrazoTotal=21
tf_dados
Frame
Console
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
The experiment run is finished. Running time: 0.4060
CAP NUM SCR
PT
00:12
14/04/2015

Dados aleatórios

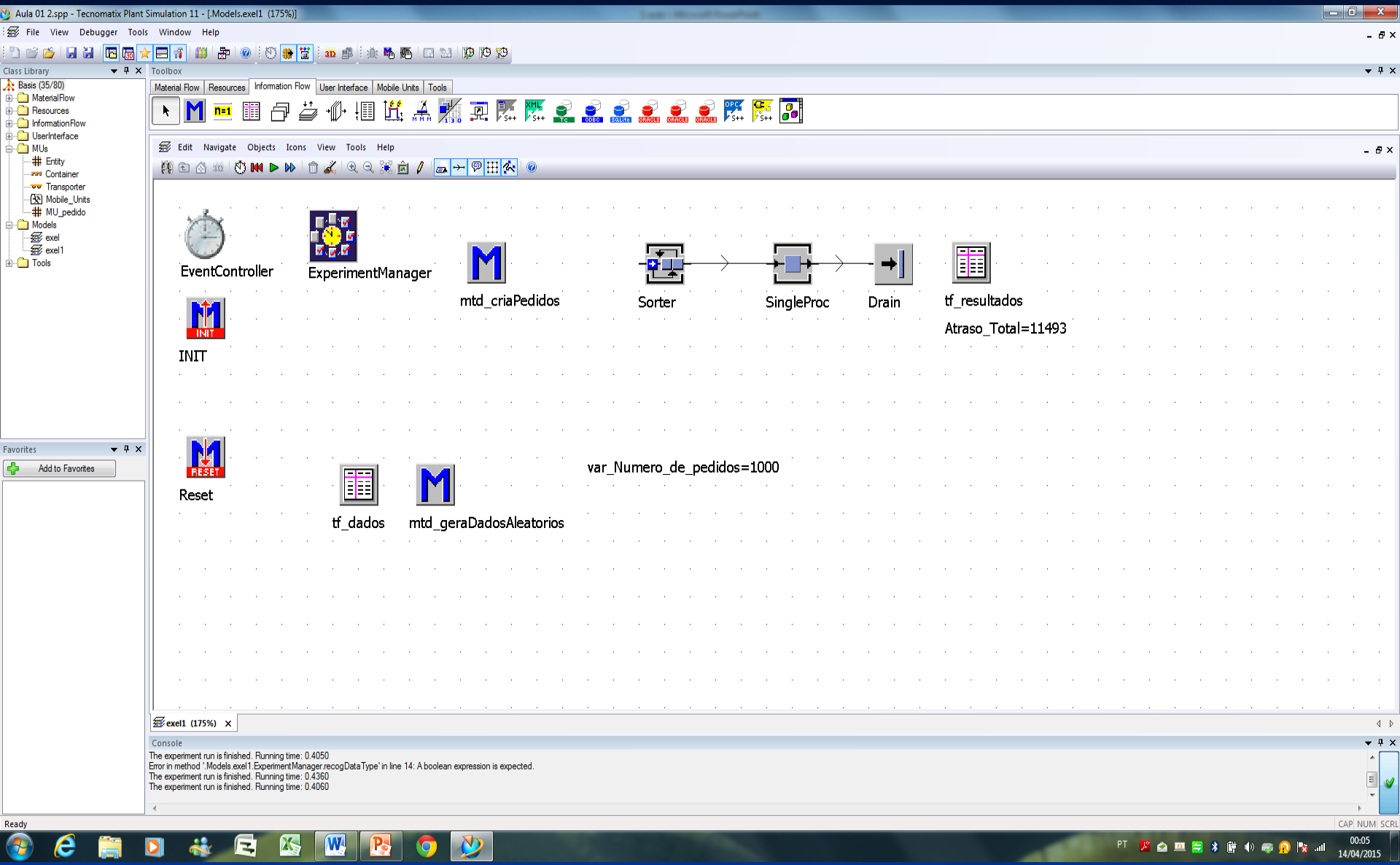
Criados pelo *Excel*

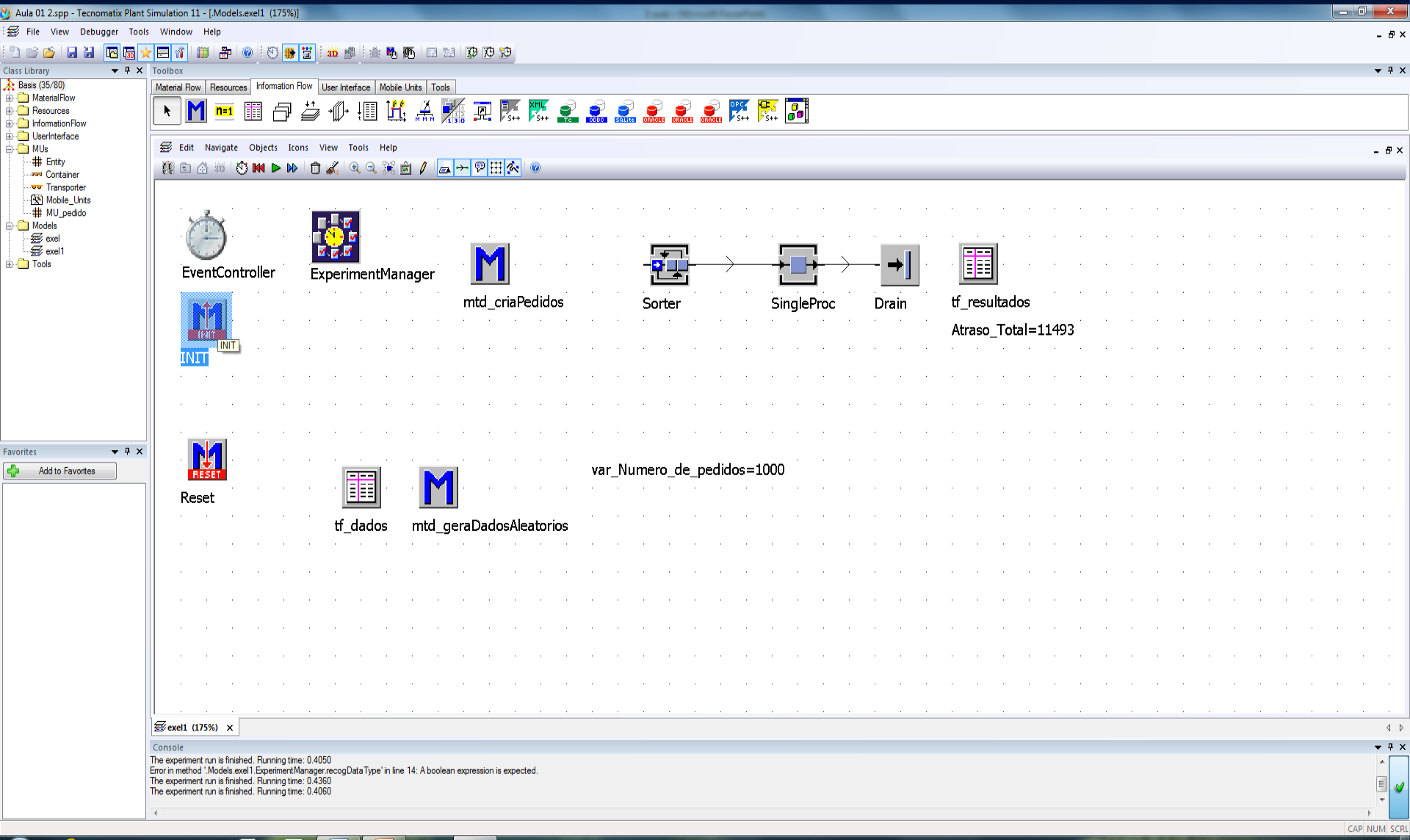


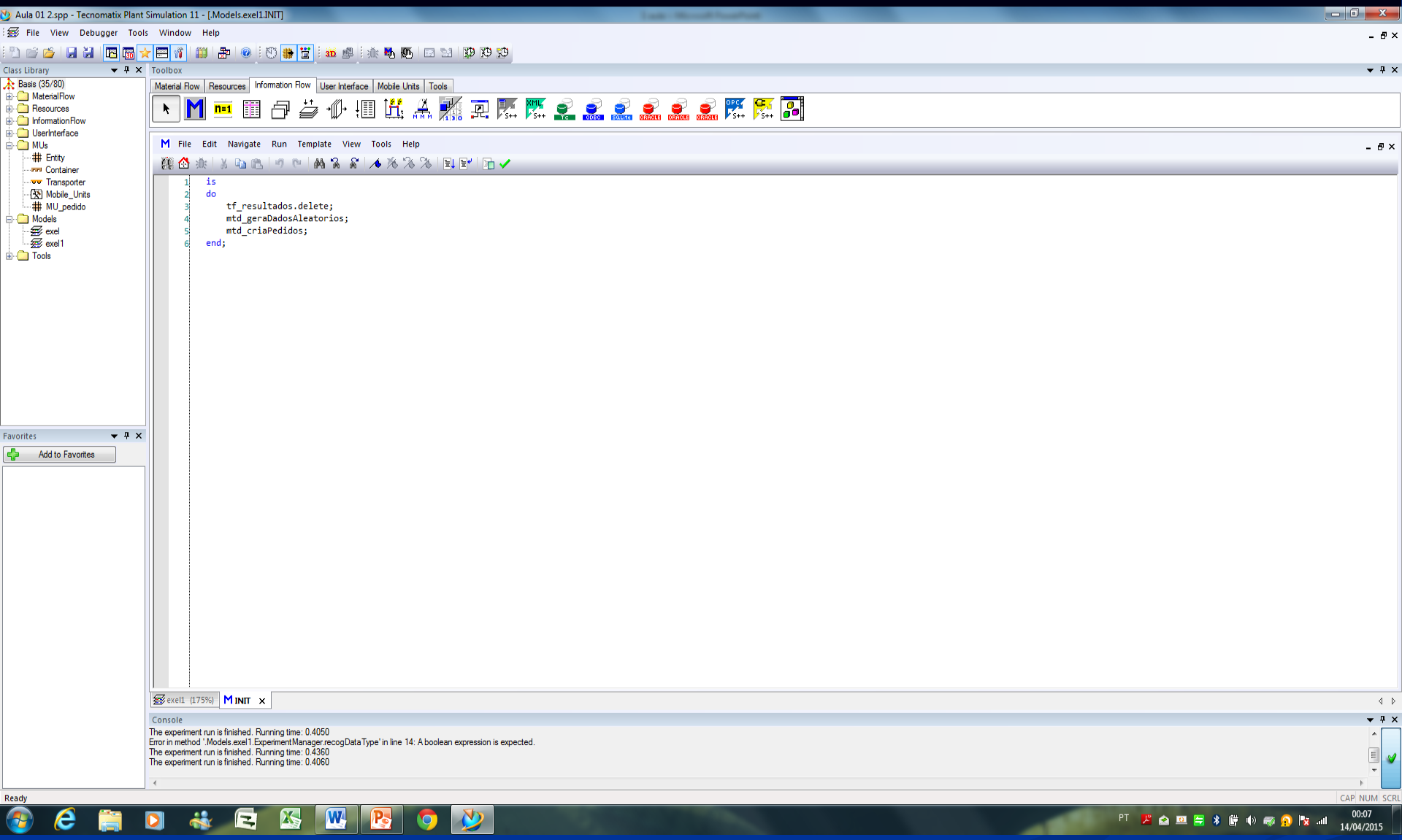

```
is
do
        tf_resultados.delete;
        mtd_importacao;
        mtd_criaPedidos;

end;
```

Dados aleatórios

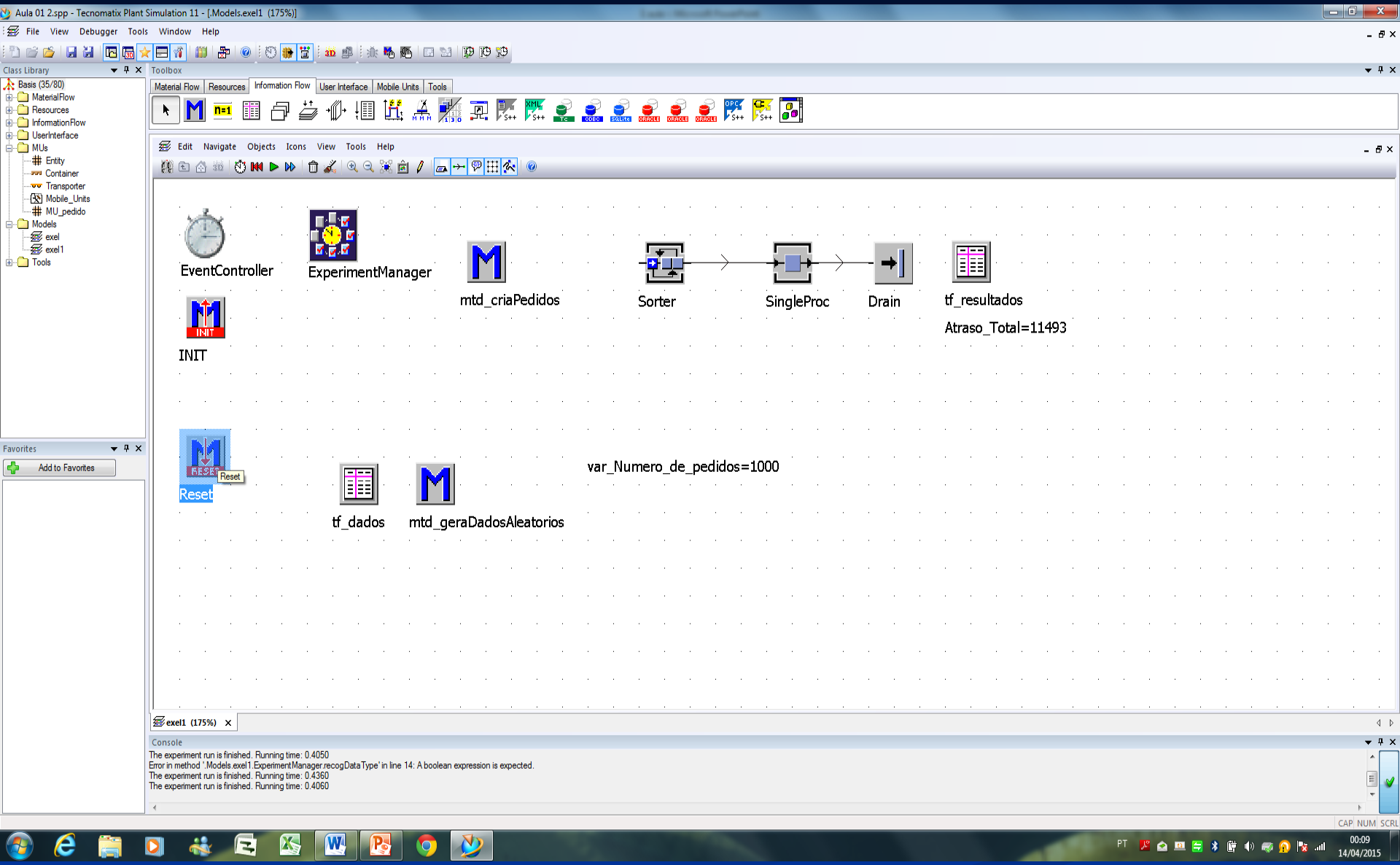
Criados pelo *Plant Simulation*

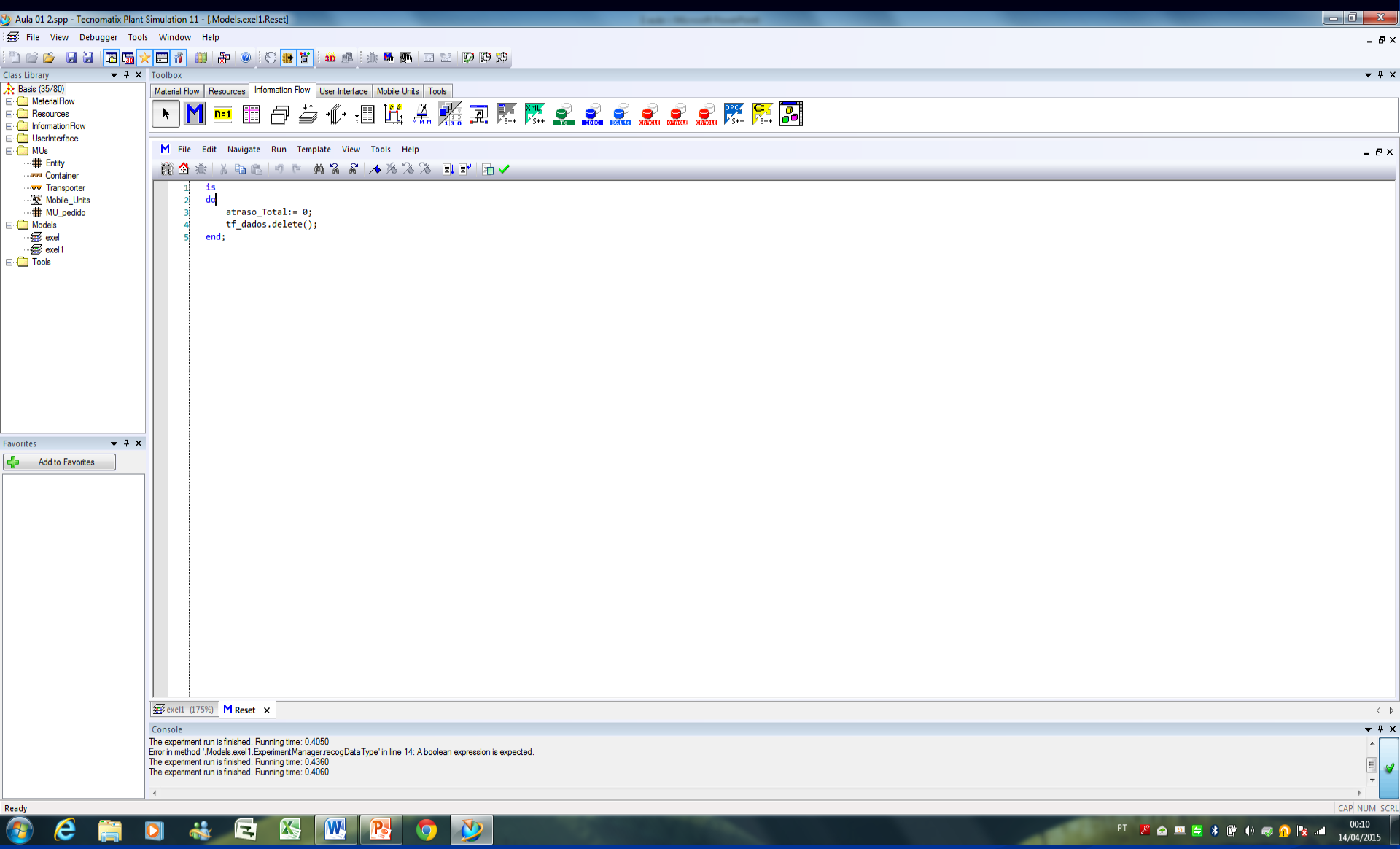

Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1 (175%)]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Edit Navigate Objects Icons View Tools Help
EventController
ExperimentManager
mtd_criaPedidos
Sorter
SingleProc
Drain
tf_resultados
Atraso_Total=11493
INIT
INIT
Reset
RESET
tf_dados
mtd_geraDadosAleatorios
var_Numero_de_pedidos=1000
exel1 (175%)
Console
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
The experiment run is finished. Running time: 0.4060
Ready
CAP NUM SCRL
PT
00:05
14/04/2015


Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1 (175%)]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Edit Navigate Objects Icons View Tools Help
EventController
ExperimentManager
mtd_criaPedidos
Sorter
SingleProc
Drain
tf_resultados
Atraso_Total=11493
INIT
Reset
tf_dados
mtd_geraDadosAleatorios
var_Numero_de_pedidos=1000
exel1 (175%)
Console
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
The experiment run is finished. Running time: 0.4060
CAP NUM SCRL
PT
00:06
14/04/2015




```
is
do
        tf_resultados.delete;
        mtd_geraDadosAleatorios;
        mtd_criaPedidos;

end;
```


Aula 01 2.spp - Tecnomatix Plant Simulation 11 - [.Models.exel1 (175%)]
File View Debugger Tools Window Help
Class Library
Basis (35/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
MU_pedido
Models
exel
exel1
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Edit Navigate Objects Icons View Tools Help
EventController
ExperimentManager
mtd_criaPedidos
Sorter
SingleProc
Drain
tf_resultados
Atraso_Total=11493
INIT
Reset
tf_dados
mtd_geraDadosAleatorios
var_Numero_de_pedidos=1000
exel1 (175%)
Console
The experiment run is finished. Running time: 0.4050
Error in method '.Models.exel1.ExperimentManager.recogDataType' in line 14: A boolean expression is expected.
The experiment run is finished. Running time: 0.4360
The experiment run is finished. Running time: 0.4060
CAP NUM SCRL
PT
00:09
14/04/2015




```
is
do
                    atraso_Total:= 0;
                    tf_dados.delete();
end;
```

Gerar dados aleatório no *Plant Simulation*

```
is
        i :integer;
do


        for i:=1 to var_Numero_de_pedidos loop
                tf_dados[1,i]:=i;
                tf_dados[2,i]:=z_uniform(1,5,60);
                tf_dados[3,i]:=z_uniform(1,0,6);
        next;
end;
```

mtd_importacao

```
is
    linhas : integer;
    i :integer;
do

    --importa dados do excel
    tf_entrada.readExcelFile("Pedidos.xlsx");
    --trata os dados
    linhas := tf_entrada.YDim - 2;

    for i := 1 to linhas loop
    tf_dados[1,i] := str_to_Num(tf_entrada[1,i+2]); --cod_produto
    tf_dados[2,i] := str_to_Num(tf_entrada[2,i+2]); --tempo_proc (min)
    tf_dados[3,i] := str_to_Num(tf_entrada[3,i+2]); --data_entrega

        next;
end;
```

mtd_geraDadosAleatorios

```
is
            i :integer;
do

        for i:=1 to var_Numero_de_pedidos loop

        tf_dados[1,i]:=i;
        tf_dados[2,i]:=z_uniform(1,5,60);
        tf_dados[3,i]:=z_uniform(1,0,6);

            next;
end;
```

https://www.plm.automation.siemens.com/pt_br/academic/resources/tecnomatix/simulation-download.cfm

HTTP://WWW.PLM.AUTOMATION.SIEMENS.COM/PT_BR/ACADEMIC/RESOURCES/TECNOMATIX/SIMULATION-THANKS.SHTML

Conceitos programação

1) Laços de repetição ⇒ For
2) Ponteiro
3) *String*
4) Inteiro
5) *Bolean* – Sim ou não
6) := ⇒ **atribui**
7) = ⇒ **compara**

EXEMPLO

SEQUENCIAMENTO

Tecla F1 – *HELP*

Tecla F5 – executa o método

√ - valida o código

Roteiro de construção do exemplo de sequenciamento

1) Importação dos dados **(importando pedido)**: construir tabela de dados: número do pedido, data devida e tempo de processamento;
2) Tratamento dos dados **(organizando pedidos)**;
3) Liberação dos dados para o *Sorter* **(transferindo pedidos para o *Sorter*)**;
4) Processamento dos pedidos no ***Single Proc***;
5) Saída – *Drain* – **resultados alcançados** (por exemplo: tempo total para o atendimento dos pedidos ou atraso total).

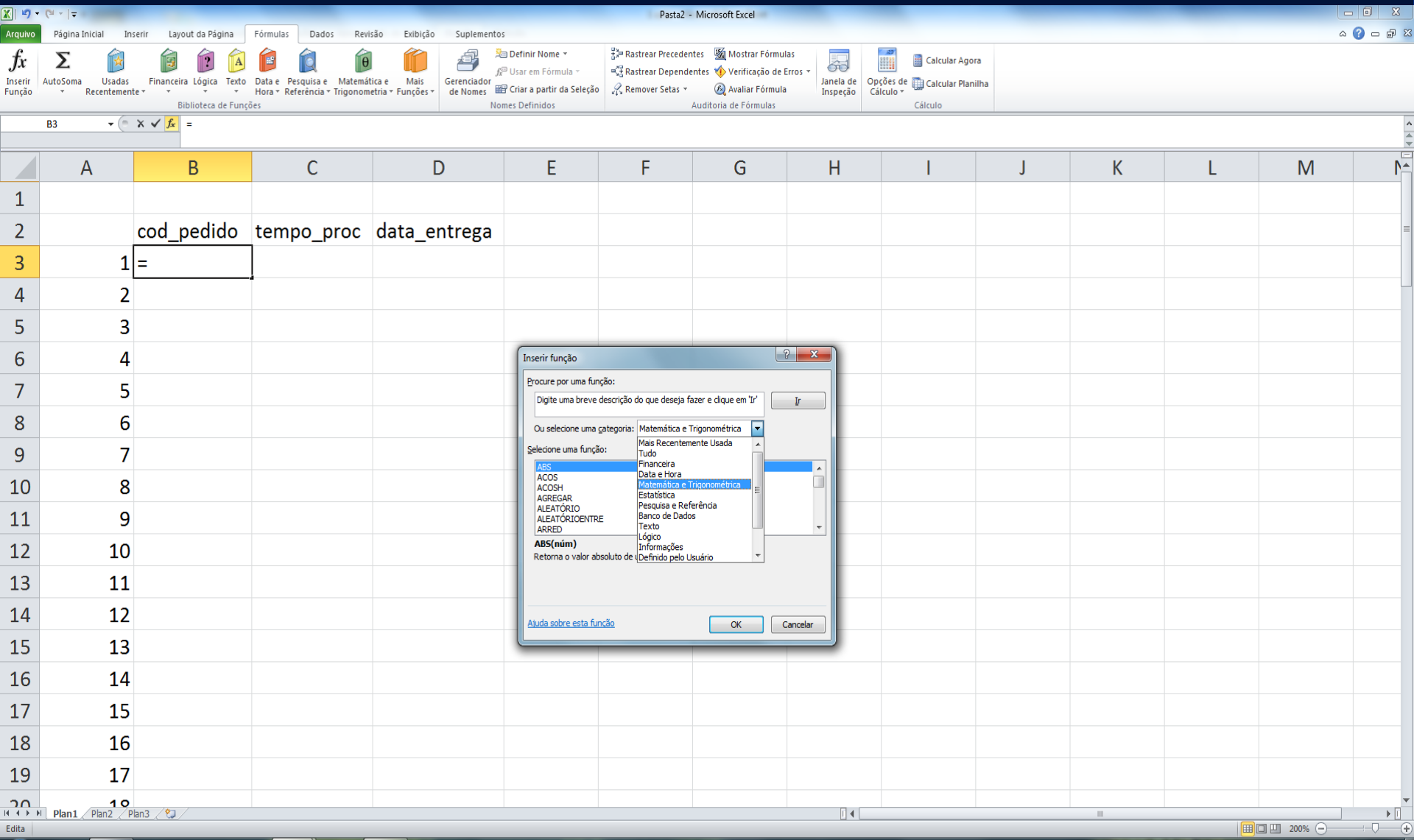
1) IMPORTAÇÃO DOS DADOS – pedidos (*cod_pedido*) – 100 pedidos

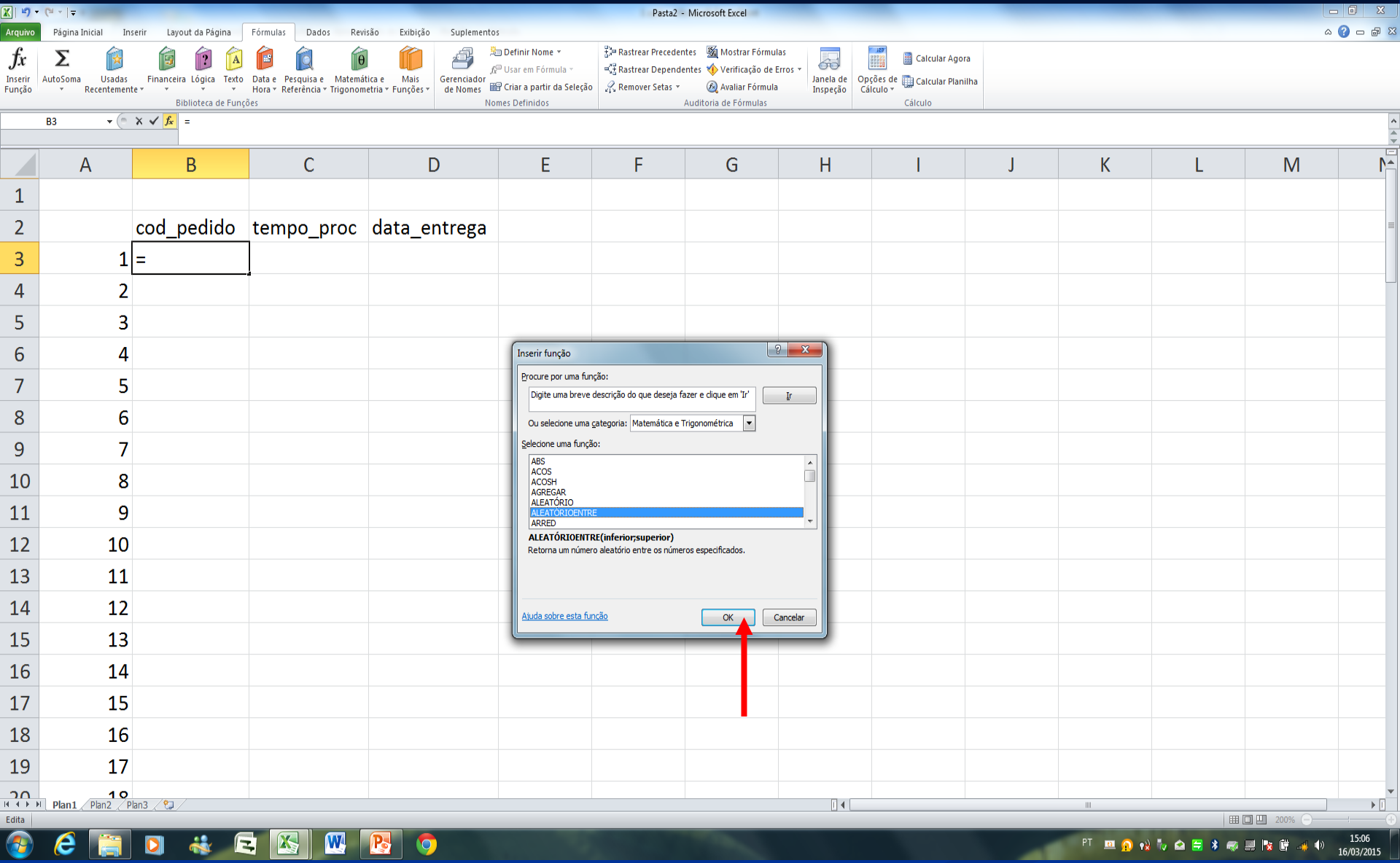
DADOS ALEATÓRIOS DO EXCELL

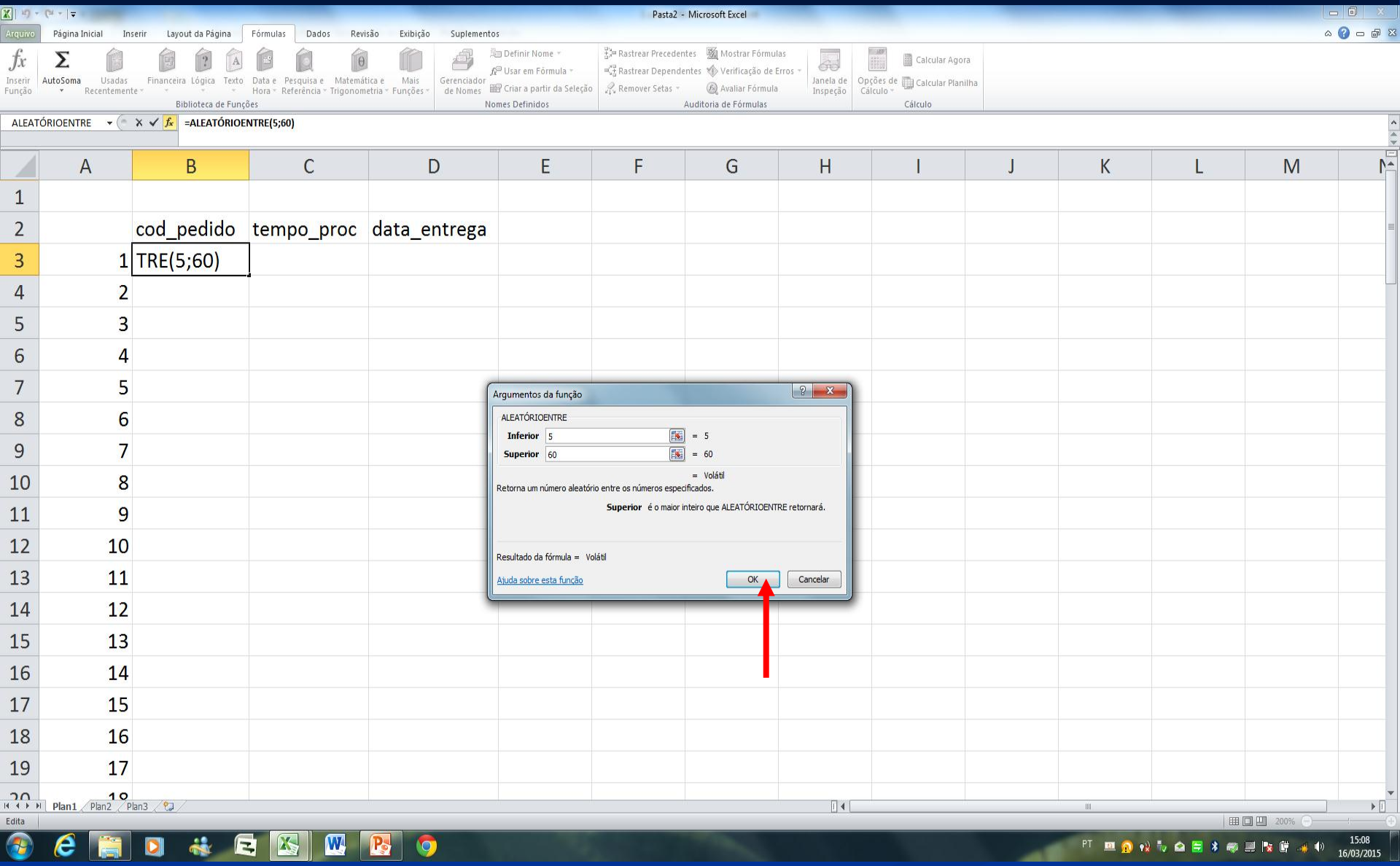
Tempos de processamento: (5 – 60) – variação por pedido de 5 a até 60 minutos (***tempo_proc***)

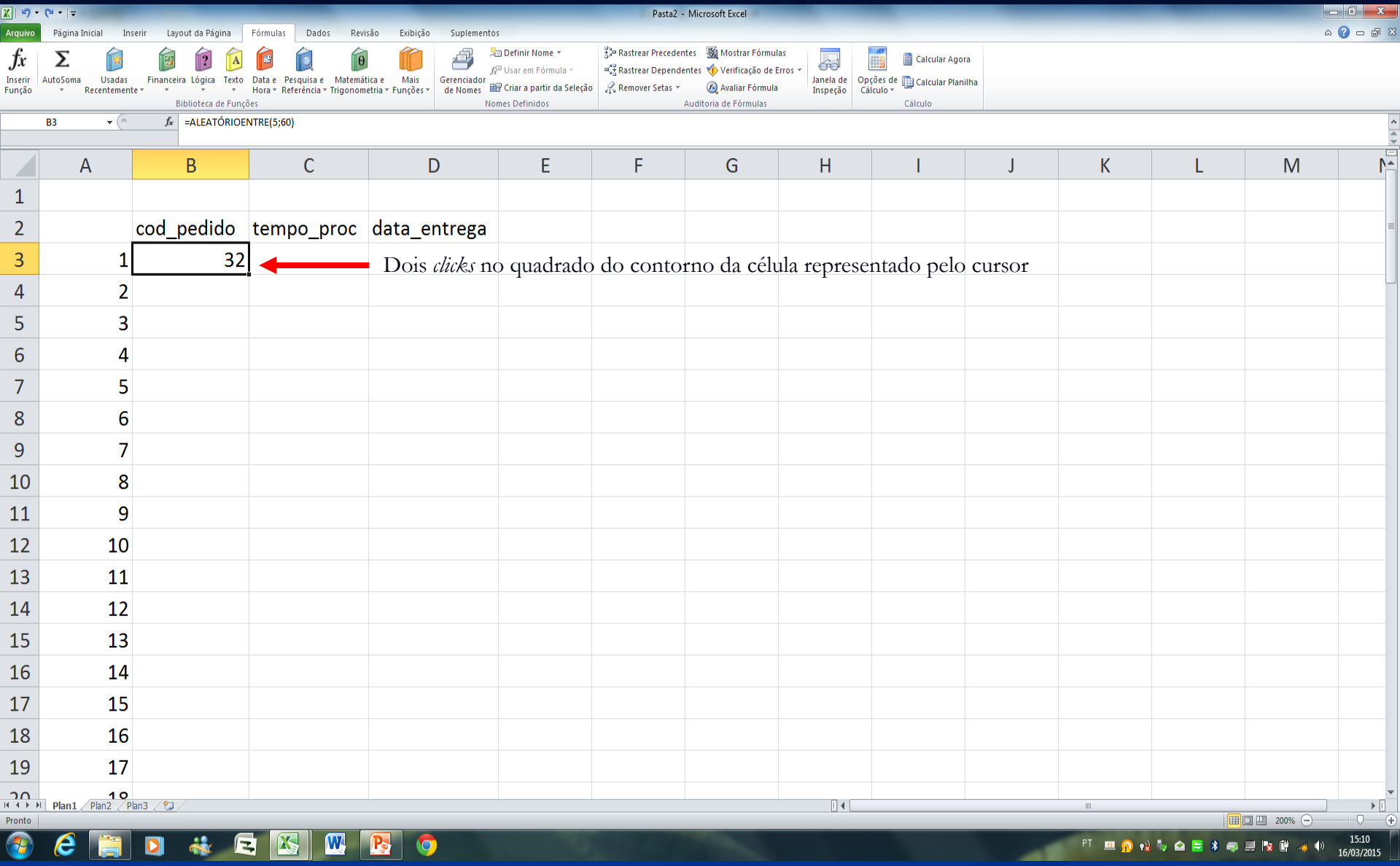
Data devida: (0 – 6) – variação por pedido de 0 (data de hoje) até 6 dias – intervalo de tempo de 1 semana (***data_entrega***)

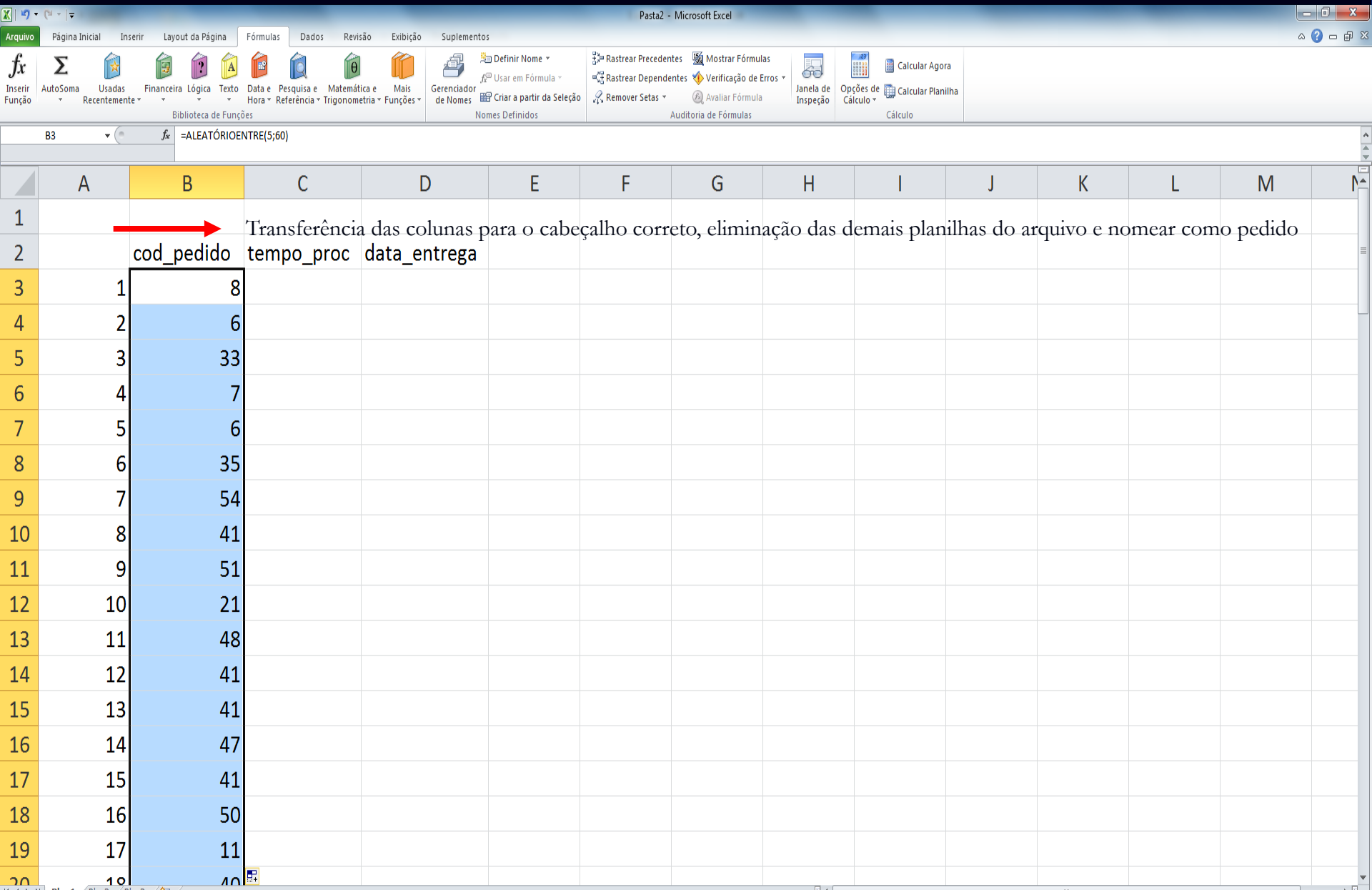
Boas práticas de programação


Pasta2 - Microsoft Excel
Arquivo
Página Inicial
Inserir
Layout da Página
Fórmulas
Dados
Revisão
Exibição
Suplementos
Inserir Função
AutoSoma
Usadas Recentemente
Financeira
Lógica
Texto
Data e Hora
Pesquisa e Referência
Matemática e Trigonometria
Mais Funções
Biblioteca de Funções
Gerenciador de Nomes
Definir Nome
Usar em Fórmula
Criar a partir da Seleção
Nomes Definidos
Rastrear Precedentes
Rastrear Dependentes
Remover Setas
Mostrar Fórmulas
Verificação de Erros
Avaliar Fórmula
Auditoria de Fórmulas
Janela de Inspeção
Opções de Cálculo
Calcular Agora
Calcular Planilha
Cálculo
B3
=
cod_pedido
tempo_proc
data_entrega
Inserir função
Procure por uma função:
Digite uma breve descrição do que deseja fazer e clique em 'Ir'
Ir
Ou selecione uma categoria:
Matemática e Trigonométrica
Selecione uma função:
Mais Recentemente Usada
Tudo
Financeira
Data e Hora
Matemática e Trigonométrica
Estatística
Pesquisa e Referência
Banco de Dados
Texto
Lógico
Informações
Definido pelo Usuário
ABS
ACOS
ACOSH
AGREGAR
ALEATÓRIO
ALEATÓRIOENTRE
ARRED
ABS(núm)
Retorna o valor absoluto de
Ajuda sobre esta função
OK
Cancelar
Plan1
Plan2
Plan3
Edita
200%
PT
15:05
16/03/2015

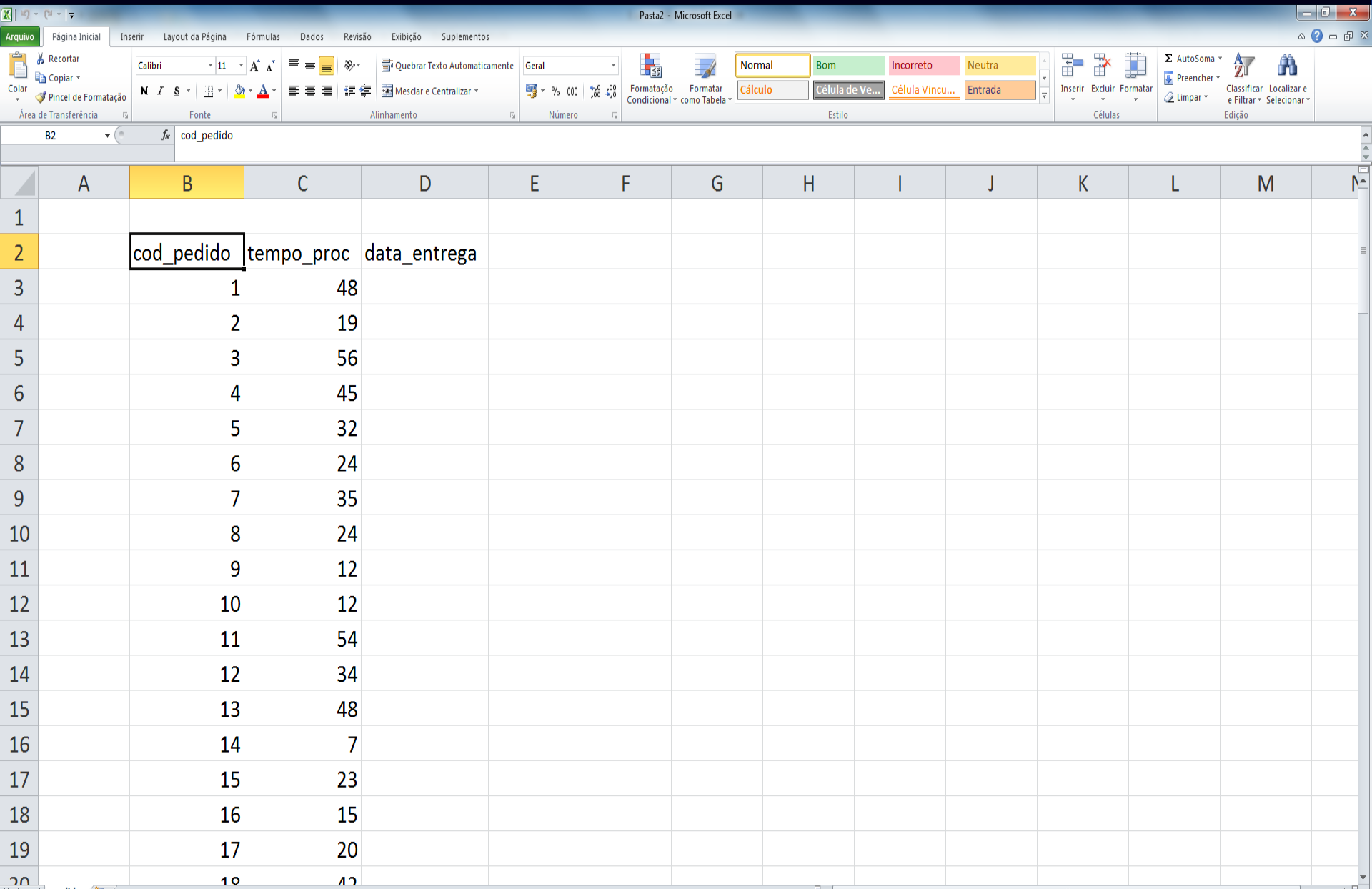

Pasta2 - Microsoft Excel
Arquivo
Página Inicial
Inserir
Layout da Página
Fórmulas
Dados
Revisão
Exibição
Suplementos
Inserir Função
AutoSoma
Usadas Recentemente
Financeira
Lógica
Texto
Data e Hora
Pesquisa e Referência
Matemática e Trigonometria
Mais Funções
Biblioteca de Funções
Gerenciador de Nomes
Definir Nome
Usar em Fórmula
Criar a partir da Seleção
Nomes Definidos
Rastrear Precedentes
Rastrear Dependentes
Remover Setas
Mostrar Fórmulas
Verificação de Erros
Avaliar Fórmula
Auditoria de Fórmulas
Janela de Inspeção
Opções de Cálculo
Calcular Agora
Calcular Planilha
Cálculo
B3
=
cod_pedido
tempo_proc
data_entrega
Inserir função
Procure por uma função:
Digite uma breve descrição do que deseja fazer e clique em 'Ir'
Ir
Ou selecione uma categoria:
Matemática e Trigonométrica
Selecione uma função:
ABS
ACOS
ACOSH
AGREGAR
ALEATÓRIO
ALEATÓRIOENTRE
ARRED
ALEATÓRIOENTRE(inferior;superior)
Retorna um número aleatório entre os números especificados.
Ajuda sobre esta função
OK
Cancelar
Plan1
Plan2
Plan3
Edita
PT
15:06
16/03/2015

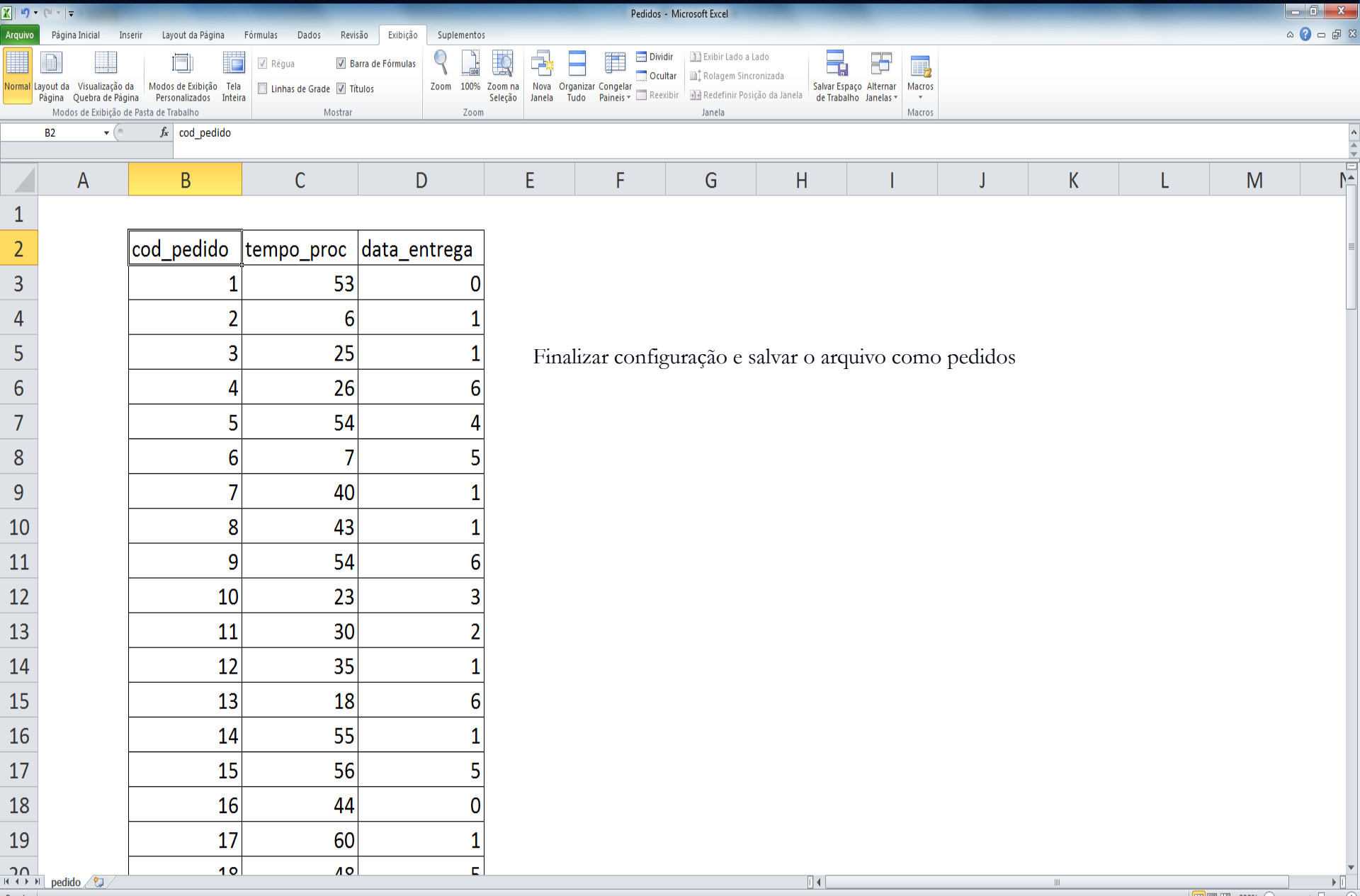

Pasta2 - Microsoft Excel
Arquivo
Página Inicial
Inserir
Layout da Página
Fórmulas
Dados
Revisão
Exibição
Suplementos
ALEATÓRIOENTRE
=ALEATÓRIOENTRE(5;60)
cod_pedido
tempo_proc
data_entrega
TRE(5;60)
Argumentos da função
ALEATÓRIOENTRE
Inferior 5 = 5
Superior 60 = 60
= Volátil
Retorna um número aleatório entre os números especificados.
Superior é o maior inteiro que ALEATÓRIOENTRE retornará.
Resultado da fórmula = Volátil
Ajuda sobre esta função
OK
Cancelar
Plan1
Plan2
Plan3
Edita


Pasta2 - Microsoft Excel
Arquivo
Página Inicial
Inserir
Layout da Página
Fórmulas
Dados
Revisão
Exibição
Suplementos
Inserir Função
AutoSoma
Usadas Recentemente
Financeira
Lógica
Texto
Data e Hora
Pesquisa e Referência
Matemática e Trigonometria
Mais Funções
Biblioteca de Funções
Gerenciador de Nomes
Definir Nome
Usar em Fórmula
Criar a partir da Seleção
Nomes Definidos
Rastrear Precedentes
Rastrear Dependentes
Remover Setas
Mostrar Fórmulas
Verificação de Erros
Avaliar Fórmula
Auditoria de Fórmulas
Janela de Inspeção
Opções de Cálculo
Calcular Agora
Calcular Planilha
Cálculo
B3
=ALEATÓRIOENTRE(5;60)
cod_pedido
tempo_proc
data_entrega
32
Dois clicks no quadrado do contorno da célula representado pelo cursor
Plan1
Plan2
Plan3
Pronto
200%
PT
15:10
16/03/2015



	A	B	C	D
1				
2		cod_pedido	tempo_proc	data_entrega
3	1	8		
4	2	6		
5	3	33		
6	4	7		
7	5	6		
8	6	35		
9	7	54		
10	8	41		
11	9	51		
12	10	21		
13	11	48		
14	12	41		
15	13	41		
16	14	47		
17	15	41		
18	16	50		
19	17	11		


Pasta2 - Microsoft Excel
Arquivo
Página Inicial
Inserir
Layout da Página
Fórmulas
Dados
Revisão
Exibição
Suplementos
Recortar
Copiar
Pincel de Formatação
Colar
Área de Transferência
Calibri
11
Fonte
Quebrar Texto Automaticamente
Mesclar e Centralizar
Alinhamento
Geral
Número
Formatação Condicional
Formatar como Tabela
Normal
Bom
Incorreto
Neutra
Cálculo
Célula de Ve...
Célula Vincu...
Entrada
Estilo
Inserir
Excluir
Formatar
Células
AutoSoma
Preencher
Limpar
Classificar e Filtrar
Localizar e Selecionar
Edição
B2
cod_pedido
A
B
C
D
E
F
G
H
I
J
K
L
M
1
2 cod_pedido tempo_proc data_entrega
3 1 48
4 2 19
5 3 56
6 4 45
7 5 32
8 6 24
9 7 35
10 8 24
11 9 12
12 10 12
13 11 54
14 12 34
15 13 48
16 14 7
17 15 23
18 16 15
19 17 20
pedido
Pronto
200%
PT
15:14
16/03/2015



cod_pedido	tempo_proc	data_entrega
1	53	0
2	6	1
3	25	1
4	26	6
5	54	4
6	7	5
7	40	1
8	43	1
9	54	6
10	23	3
11	30	2
12	35	1
13	18	6
14	55	1
15	56	5
16	44	0
17	60	1

Finalizar configuração e salvar o arquivo como pedidos

Versão de Hodgson

Algoritmo de Moore

Procedimento – versão 01

1) Passo 1. Sequencie as tarefas de acordo com a regra EDD ((*Earliest Due Date* – são programadas primeiro as ordens mais próximas às datas de entrega)).
2) Passo 2. Identifique a primeira tarefa atrasada, por exemplo i(*l*), na sequência corrente. Se não existe tarefa atrasada, vá para o passo 4.
3) Passo 3. Identifique a tarefa na subsequência (i(1), i(2),…,i(*l*)) com o maior tempo de processamento e retire-a da sequência corrente. Retorne ao passo 2 com a sequência corrente.
4) Passo 4. Forme uma sequência ótima a partir da sequência corrente e adicione à mesma as tarefas retiradas no passo 3 que podem ser sequenciadas em qualquer ordem.

Nota: As tarefas atrasadas são aquelas retiradas no passo 3.

Procedimento – versão 02

1) Passo 1 – Ordene as tarefas segundo a regra EDD (*Earliest Due Date* – são programadas primeiro as ordens mais próximas às datas de entrega). Esta ordenação é denominada sequência atual.
2) Passo 2 – Na sequência atual, identifique a primeira tarefa com atraso. Esta tarefa e as suas precedentes determinam uma subsequência de confronto. Se tal tarefa for identificada, vá para o passo 3. Caso contrário, a sequência ótima é a sequência atual seguida da sequência das tarefas removidas (no passo 3).
3) Passo 3 – Remova a tarefa como maior tempo de processamento entre as tarefas da subsequência de confronto e coloque-a na sequência das tarefas removidas. A sequência total é formada pela sequência atual e sequência das tarefas removidas. Vá para o passo 2.

Problema 01 – 6 / 1 / n_T

Tarefa	1	2	3	4	5	6
Data de Entrega	15	6	9	23	20	30
Tempo de Processamento	10	3	4	8	10	6

Problema 02 – 8 / 1 / n_T

Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9

Número total de tarefas atrasadas $\Rightarrow n_T$

Solução

Problema 01

Solução – (Passos 1 e 2)

Problema 01 – 6 / 1 / n_T

Dados						
Tarefa	1	2	3	4	5	6
Data de Entrega	15	6	9	23	20	30
Tempo de Processamento	10	3	4	8	10	6
Intervalo de tempo	**+ 5**	**+ 3**	**+ 5**	**+ 15**	**+ 10**	**+ 24**
Passos 1 e 2						
Sequência Corrente	2	3	1	5	4	6
Data de Entrega	6	9	15	20	23	30
Tempo de Processamento	3	4	10	10	8	6
Instante de Término	3	7	17	27	35	41
Há atraso?	**+ 3**	**+ 2**	**- 2**	**- 7**	**- 12**	**- 11**
Resultado: quatro tarefas atrasadas $\Rightarrow$ 1 – 5 – 4 – 6						

Análise

- A tarefa 1 é a primeira tarefa atrasada e na subsequência (2, 3, 1) é a que possui maior tempo de processamento. Retire a tarefa 1 e vá para o passo 2.

Solução – (Retirada da Tarefa 1)

Problema 01 – 6 / 1 / nT

Dados						
Tarefa	1	2	3	4	5	6
Data de Entrega	15	6	9	23	20	30
Tempo de Processamento	10	3	4	8	10	6
Intervalo de tempo	**+ 5**	**+ 3**	**+ 5**	**+ 15**	**+ 10**	**+ 24**
Passos 1 e 2						
Sequência Corrente	2	3		5	4	6
Data de Entrega	6	9		20	23	30
Tempo de Processamento	3	4		10	8	6
Instante de Término	3	7		17	25	31
Há atraso?	**+ 3**	**+ 2**		**+ 3**	**- 2**	**- 1**
Resultado: quatro tarefas atrasadas $\Rightarrow$ 1 – 5 – 4 – 6						

Solução – (Retirada da Tarefa 1)

Problema 01 – 6 / 1 / nT

Dados						
Tarefa	1	2	3	4	5	6
Data de Entrega	15	6	9	23	20	30
Tempo de Processamento	10	3	4	8	10	6
Intervalo de tempo	**+ 5**	**+ 3**	**+ 5**	**+ 15**	**+ 10**	**+ 24**
Passos 1 e 2						
Sequência Corrente	2	3	5	4	6	
Data de Entrega	6	9	20	23	30	
Tempo de Processamento	3	4	10	8	6	
Instante de Término	3	7	17	25	31	
Resultado: Tarefas Retiradas – 1						

Análise

- A tarefa 4 é a primeira tarefa atrasada na sequência e na subsequência (2, 3, 5, 4), a tarefa 5 tem o maior tempo de processamento. Retire a tarefa 5 e retorne ao passo 2.

Solução – (Retirada da Tarefa 5)

Problema 01 – 6 / 1 / n_T

Dados						
Tarefa	1	2	3	4	5	6
Data de Entrega	15	6	9	23	20	30
Tempo de Processamento	10	3	4	8	10	6
Intervalo de tempo	**+ 5**	**+ 3**	**+ 5**	**+ 15**	**+ 10**	**+ 24**
Passos 1 e 2						
Sequência Corrente	2	3			4	6
Data de Entrega	6	9			23	30
Tempo de Processamento	3	4			8	6
Instante de Término	3	7			15	21
Há atraso?	**+ 3**	**+ 2**			**+ 8**	**+ 9**
Resultado: três tarefas atrasadas $\Rightarrow$ 5 – 4 – 6						

Solução – (Retirada da Tarefa 5)

Problema 01 – 6 / 1 / nT

Dados						
Tarefa	1	2	3	4	5	6
Data de Entrega	15	6	9	23	20	30
Tempo de Processamento	10	3	4	8	10	6
Intervalo de tempo	**+ 5**	**+ 3**	**+ 5**	**+ 15**	**+ 10**	**+ 24**
Passos 1 e 2						
Sequência Corrente	2	3	4	6		
Data de Entrega	6	9	23	30		
Tempo de Processamento	3	4	8	6		
Instante de Término	3	7	15	21		
Resultado: Tarefas Retiradas – 1 – 5						

Análise

- Passo 4. Sequências ótimas: (2, 3, 4, 6, 1, 5) e (2, 3, 4, 6, 5, 1).

Solução – (sequência – 2 – 3 – 4 – 6 – 1 – 5)
Problema 01 – 6 / 1 / nT

Dados						
Tarefa	**1**	2	3	4	**5**	6
Data de Entrega	**15**	6	9	23	**20**	30
Tempo de Processamento	**10**	3	4	8	**10**	6
Intervalo de tempo	**+ 5**	**+ 3**	**+ 5**	**+ 15**	**+ 10**	**+ 24**
Passos 1 e 2						
Sequência Corrente	2	3	4	6	**1**	**5**
Data de Entrega	6	9	23	30	**15**	**20**
Tempo de Processamento	3	4	8	6	**10**	**10**
Instante de Término	3	7	15	21	**31**	**41**
Resultado: Tarefas Retiradas – 1 – 5						

Solução – (sequência – 2 – 3 – 4 – 6 – 5 – 1)
Problema 01 – 6 / 1 / nT

Dados						
Tarefa	**1**	2	3	4	**5**	6
Data de Entrega	**15**	6	9	23	**20**	30
Tempo de Processamento	**10**	3	4	8	**10**	6
Intervalo de tempo	**+ 5**	**+ 3**	**+ 5**	**+ 15**	**+ 10**	**+ 24**
Passos 1 e 2						
Sequência Corrente	2	3	4	6	**5**	**1**
Data de Entrega	6	9	23	30	**20**	**15**
Tempo de Processamento	3	4	8	6	**10**	**10**
Instante de Término	3	7	15	21	**31**	**41**
Resultado: Tarefas Retiradas – 1 – 5						

Solução

Problema 02

Todas as tarefas em função do tempo de processamento e *due dates* individuais partem do pressuposto que já estão atrasadas. O objetivo é minimizar o atraso.

Procedimento – versão 01

1) Passo 1. Sequencie as tarefas de acordo com a regra EDD ((*Earliest Due Date* – são programadas primeiro as ordens mais próximas às datas de entrega)).
2) Passo 2. Identifique a primeira tarefa atrasada, por exemplo i(*l*), na sequência corrente. Se não existe tarefa atrasada, vá para o passo 4.
3) Passo 3. Identifique a tarefa na subsequência (i(1), i(2),…,i(*l*)) com o maior tempo de processamento e retire-a da sequência corrente. Retorne ao passo 2 com a sequência corrente.
4) Passo 4. Forme uma sequência ótima a partir da sequência corrente e adicione à mesma as tarefas retiradas no passo 3 que podem ser sequenciadas em qualquer ordem.

Nota: As tarefas atrasadas são aquelas retiradas no passo 3.

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Intervalo de tempo	**- 25**	**- 14**	**- 8**	**- 7**	**- 2**	**- 17**	**- 21**	**- 3**
Passos 1 e 2								
Sequência Corrente	4	3	5	2	8	7	6	1
Data de Entrega	1	3	4	6	6	7	8	10
Tempo de Processamento	8	11	6	**20**	**9**	28	25	35
Instante de Término	8	19	25	45	54	82	107	142
Resultado: todas as tarefas atrasadas $\Rightarrow$ 4 – 3 – 5 – 2 – 8 – 7 – 6 – 1								

O procedimento de Moore não se aplica neste caso.

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / nT

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Intervalo de tempo	**- 25**	**- 14**	**- 8**	**- 7**	**- 2**	**- 17**	**- 21**	**- 3**
Passos 1 e 2								
Sequência Corrente	4	3	5	8	2	7	6	1
Data de Entrega	1	3	4	6	6	7	8	10
Tempo de Processamento	8	11	6	**9**	**20**	28	25	35
Instante de Término	8	19	25	34	54	82	107	142
Intervalo de tempo	**- 7**	**- 16**	**- 8**	**- 7**	**- 2**	**- 17**	**- 21**	**- 3**
Resultado: todas as tarefas atrasadas $\Rightarrow$ 4 – 3 – 5 – 2 – 8 – 7 – 6 – 1								

O procedimento de Moore não se aplica neste caso.

Análise

- A tarefa 4 é a primeira tarefa atrasada e na subsequência (4) é a que possui maior tempo de processamento, consequentemente, pois não há comparação. Retire a tarefa 4 e vá para o passo 2.

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente		3	5	2	8	7	6	1
Data de Entrega		3	4	6	6	7	8	10
Tempo de Processamento		11	6	20	9	28	25	35
Instante de Término		19	25	45	54	82	107	142
Resultado: todas as tarefas atrasadas $\Rightarrow$ 4 – 3 – 5 – 2 – 8 – 7 – 6 – 1								

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente	3	5	2	8	7	6	1	
Data de Entrega	3	4	6	6	7	8	10	
Tempo de Processamento	11	6	20	9	28	25	35	
Instante de Término	11	17	37	46	74	99	134	
Resultado: Tarefas Retiradas ⇒ 4								

Análise

- A tarefa 3 é a primeira tarefa atrasada na sequência e na subsequência (3), a tarefa 3 tem o maior tempo de processamento, não havendo comparação. Retire a tarefa 3 e retorne ao passo 2.

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente			5	2	8	7	6	1
Data de Entrega			4	6	6	7	8	10
Tempo de Processamento			6	20	9	28	25	35
Instante de Término			6	26	35	63	88	113
Resultado: todas as tarefas atrasadas $\Rightarrow$ 3 – 5 – 2 – 8 – 7 – 6 – 1								

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente	5	2	8	7	6	1		
Data de Entrega	4	6	6	7	8	10		
Tempo de Processamento	6	20	9	28	25	35		
Instante de Término	6	26	35	63	88	113		
Resultado: Tarefas Retiradas ⇒ 4 – 3								

Análise

- A tarefa 5 é a primeira tarefa atrasada na sequência e na subsequência (5), a tarefa 5 tem o maior tempo de processamento, não havendo comparação. Retire a tarefa 5 e retorne ao passo 2.

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente				2	8	7	6	1
Data de Entrega				6	6	7	8	10
Tempo de Processamento				20	9	28	25	35
Instante de Término				20	29	57	72	107
Resultado: todas as tarefas atrasadas $\Rightarrow$ 2 – 8 – 7 – 6 – 1								

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente	2	8	7	6	1			
Data de Entrega	6	6	7	8	10			
Tempo de Processamento	20	9	28	25	35			
Instante de Término	20	29	57	72	107			
Resultado: Tarefas Retiradas ⇒ 4 – 3 – 5								

Análise

- A tarefa 2 é a primeira tarefa atrasada na sequência e na subsequência (2), a tarefa 2 tem o maior tempo de processamento, não havendo comparação. Retire a tarefa 2 e retorne ao passo 2.

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente					8	7	6	1
Data de Entrega					6	7	8	10
Tempo de Processamento					9	28	25	35
Instante de Término					29	57	72	107
Resultado: todas as tarefas atrasadas $\Rightarrow$ 8 – 7 – 6 – 1								

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente	8	7	6	1				
Data de Entrega	6	7	8	10				
Tempo de Processamento	9	28	25	35				
Instante de Término	9	37	62	97				
Resultado: Tarefas Retiradas ⇒ 4 – 3 – 5 – 2								

Análise

- A tarefa 8 é a primeira tarefa atrasada na sequência e na subsequência (8), a tarefa 8 tem o maior tempo de processamento, não havendo comparação. Retire a tarefa 8 e retorne ao passo 2.

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente						7	6	1
Data de Entrega						7	8	10
Tempo de Processamento						28	25	35
Instante de Término						28	53	88
Resultado: todas as tarefas atrasadas $\Rightarrow$ 7 – 6 – 1								

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente	7	6	1					
Data de Entrega	7	8	10					
Tempo de Processamento	28	25	35					
Instante de Término	28	53	88					
Resultado: Tarefas Retiradas ⇒ 4 – 3 – 5 – 2 – 8								

Análise

- A tarefa 7 é a primeira tarefa atrasada na sequência e na subsequência (7), a tarefa 7 tem o maior tempo de processamento, não havendo comparação. Retire a tarefa 7 e retorne ao passo 2.

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente							6	1
Data de Entrega							8	10
Tempo de Processamento							25	35
Instante de Término							25	60
Resultado: todas as tarefas atrasadas $\Rightarrow$ 6 – 1								

Solução – (Passos 1 e 2)
Problema 02 – 6 / 1 / n_T

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Sequência Corrente	6	1						
Data de Entrega	8	10						
Tempo de Processamento	25	35						
Instante de Término	25	60						
Resultado: Tarefas Retiradas ⇒ 4 – 3 – 5 – 2 – 8 – 7								

Análise

- Como todas as tarefas relacionadas independente da sequência encontram-se no *status* de atraso em função da *Due Date* e do *Process Time*, o procedimento de Moore não se aplica. Sem avaliar algum outro procedimento que possa ser aplicado o mais provável é a sequência com base no tempo de processamento crescente de modo a minimizar o atraso, e não eliminá-lo.

Solução – (Sequência Final)

Problema 02 – 6 / 1 / nT

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Tempo de Processamento crescente	5	4	8	3	2	6	7	1
Data de Entrega	4	1	6	3	6	8	7	10
Tempo de Processamento	6	8	9	11	20	25	28	35
Instante de Término	6	14	23	34	54	79	107	142
Há atraso?	**- 2**	**- 13**	**- 17**	**- 31**	**- 48**	**- 71**	**- 100**	**- 132**

Solução – (Sequência Final)

Problema 02 – 6 / 1 / nT

Dados								
Tarefa	1	2	3	4	5	6	7	8
Data de Entrega	10	6	3	1	4	8	7	6
Tempo de Processamento	35	20	11	8	6	25	28	9
Passos 1 e 2								
Data de entrega crescente	4	3	5	8	2	7	6	1
Data de Entrega	1	3	4	6	6	7	8	10
Tempo de Processamento	8	11	6	9	20	28	25	35
Instante de Término	8	19	25	34	54	82	107	142
Há atraso?	**- 7**	**- 16**	**- 21**	**- 28**	**- 48**	**- 75**	**- 99**	**- 132**

Procedimento de Moore

Plant Simulation

Random

.Models.random. (random ≅ Frame)

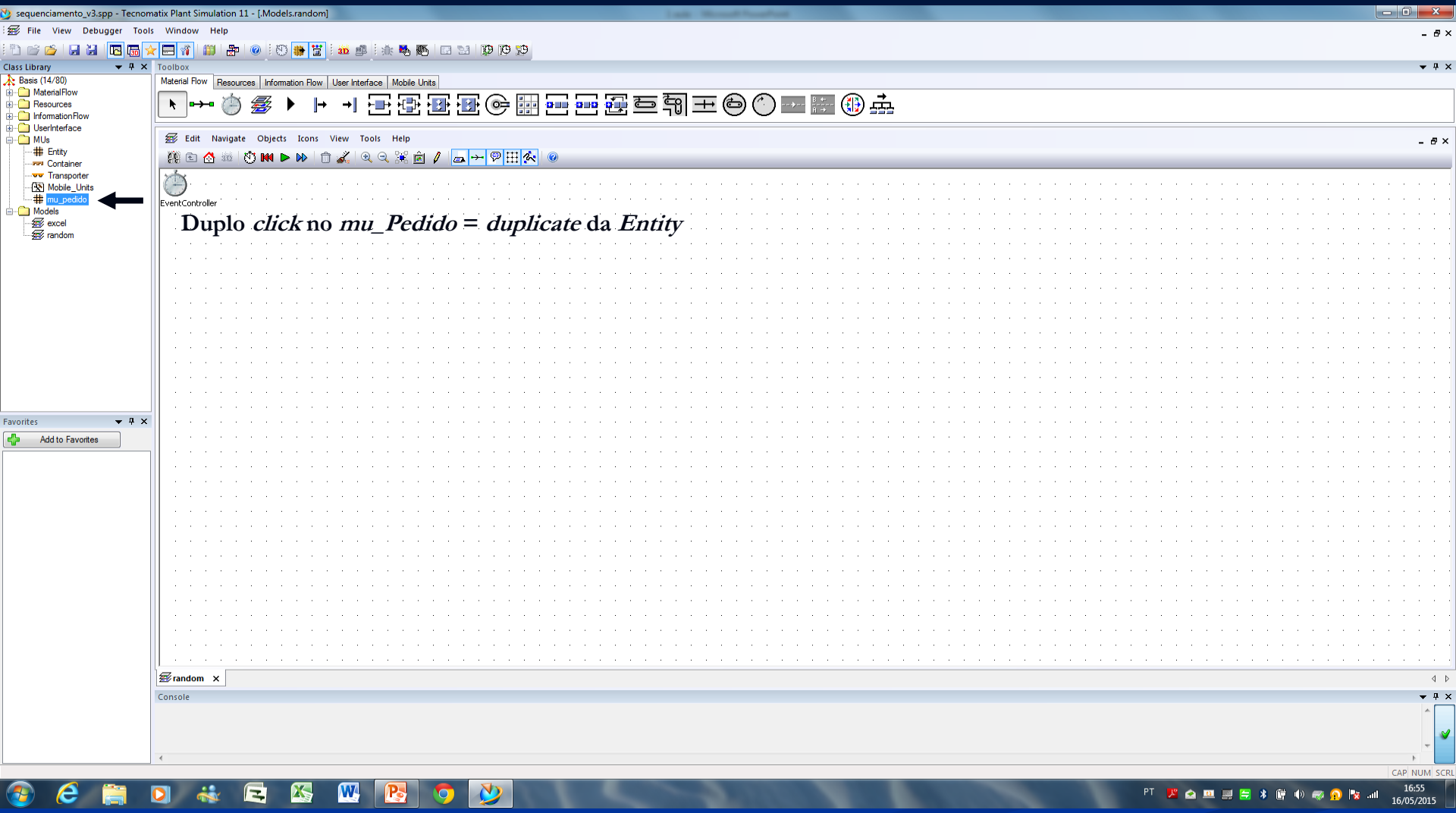
Variáveis do pedido no processo

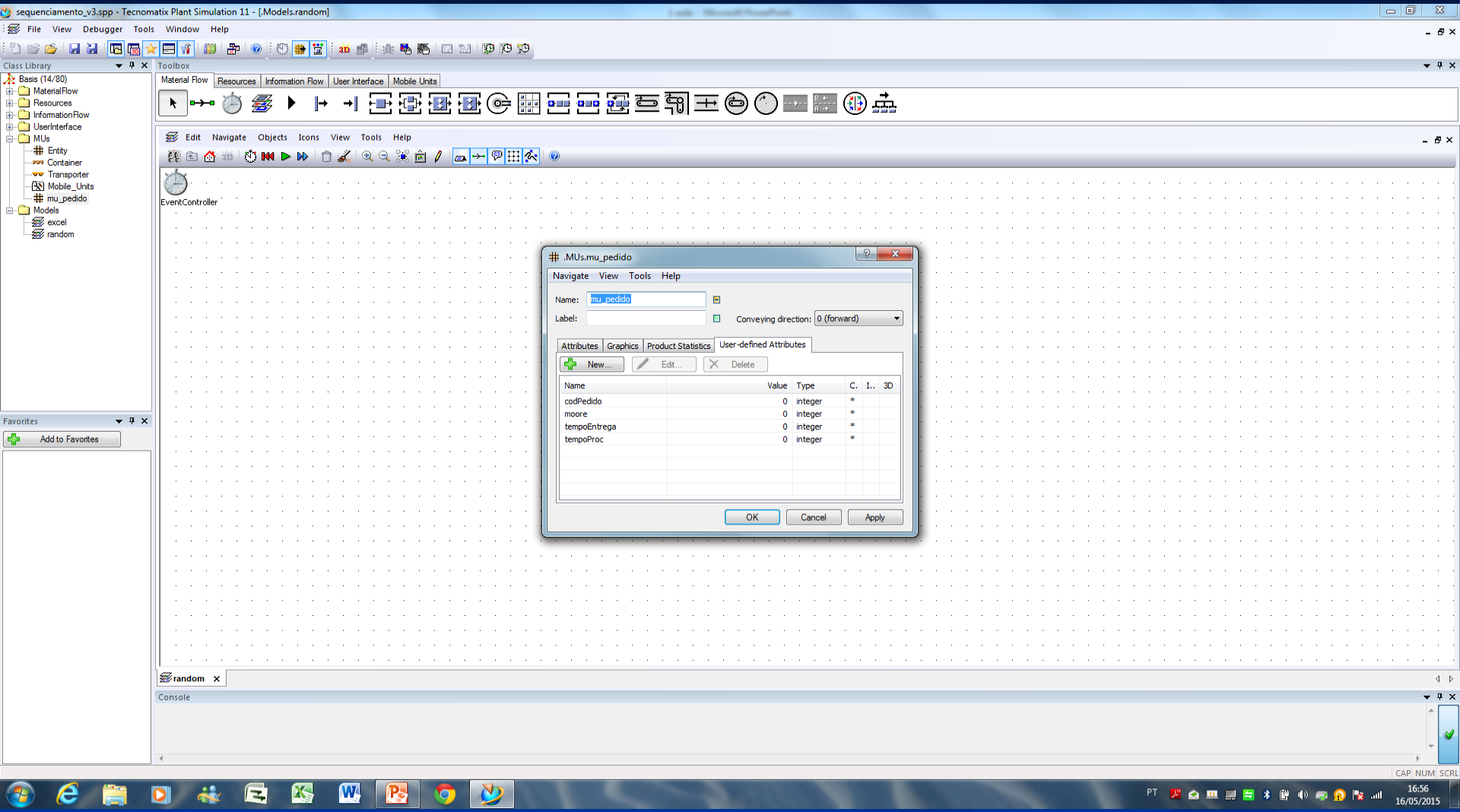
Variáveis	***Entity* ⇒ mu.pedido**	***Table* *tf_pedido*s**	**Método 3º método mtd_cria_pedidos**	***Table* tf.resultados_MOORE**
Código do Pedido	codPedido	Cod_Pedido	pedido.codPedido := tf_pedidos["**Cod_Pedido**",i];	CodPedido
Tempo de Processamento	tempoProc	Tempo_Processamento	**pedido.tempoProc** := tf_pedidos["Tempo_Processamento",i];	tempoPrevisto
Tempo de Entrega	tempoEntrega	Tempo_Entrega	***pedido.tempoEntrega*** := tf_pedidos["Tempo_Entrega",i];	tempoentregue
Moore	moore	----------	----------	TempoAtraso

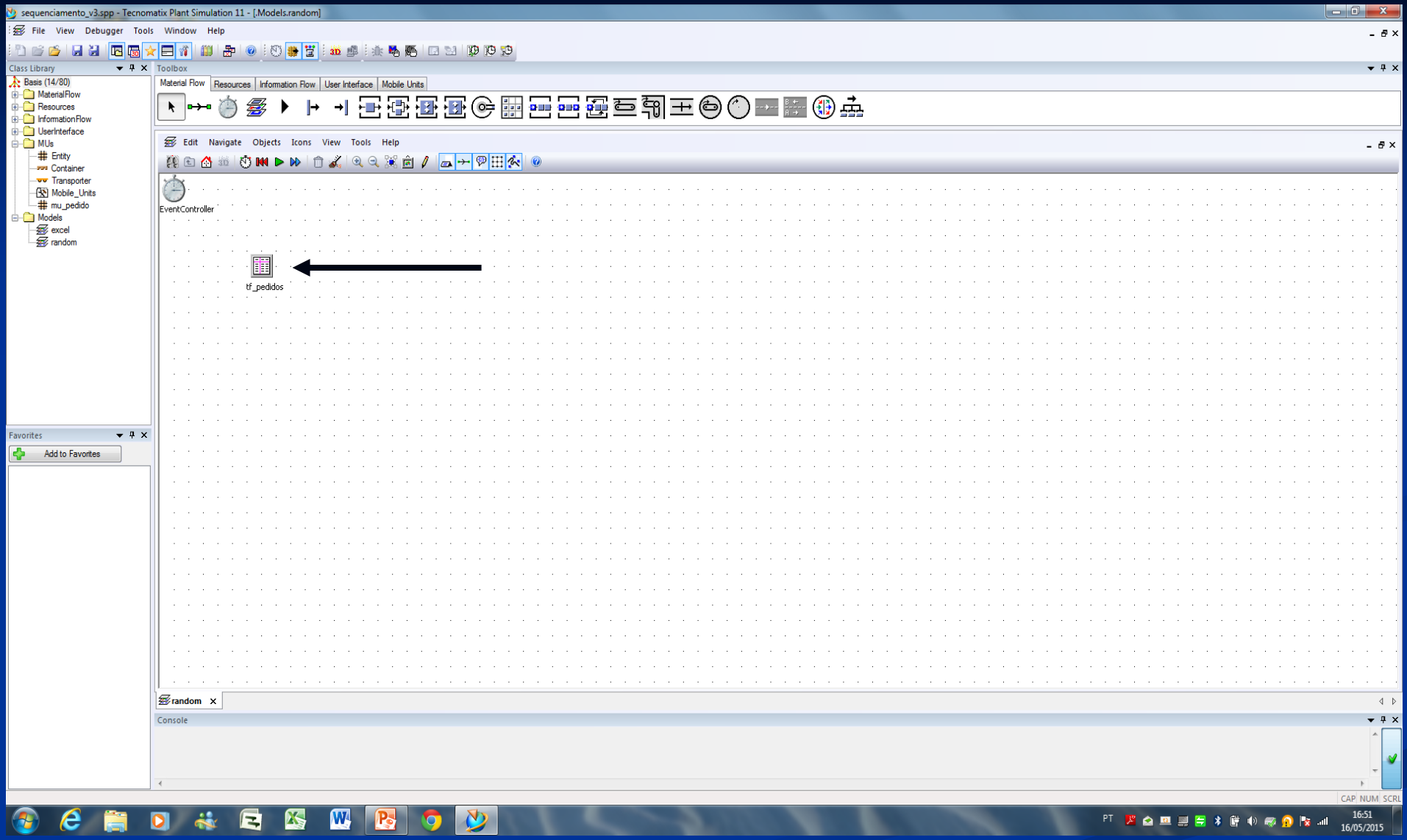
Variáveis do pedido

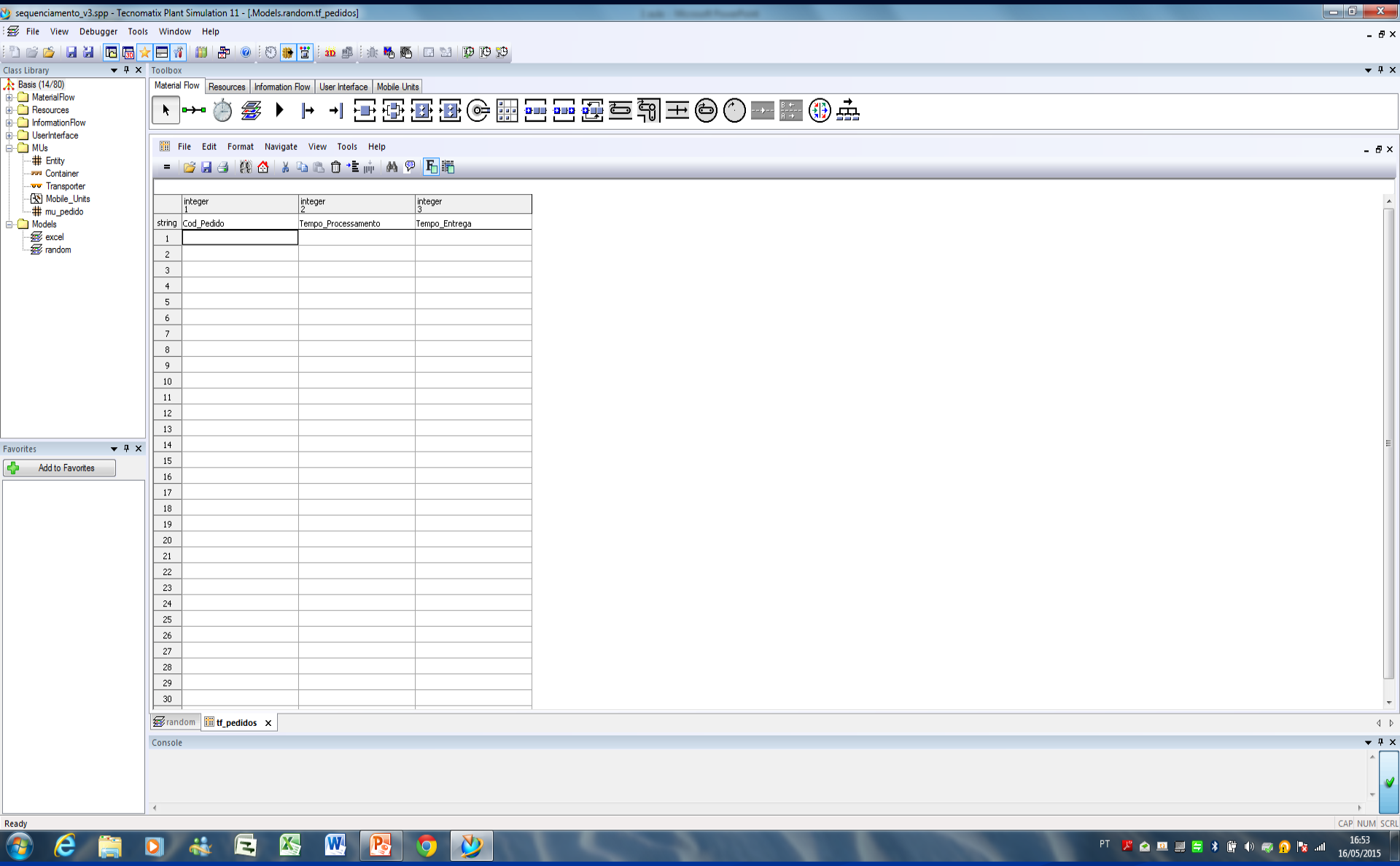
1) Código do Pedido
2) Tempo de Processamento
3) Tempo de Entrega – *Due Date*

Variável de entrada (*Input*)	***Table* Conotação Inicial *Input – SimTalk***	**Método**		**Atribuição *SimTalk* no modelo**
		Identificação do método	**Descrição**	
Código do Pedido	cod_Pedido (**1ª Coluna - *Table***)			
Tempo de processamento	Tempo_Processamento (**2ª Coluna - *Table***)			
Tempo de Entrega	Tempo_Entrega (**3ª Coluna - *Table***)			


sequenciamento_v3.spp - Tecnomatix Plant Simulation 11 - [.Models.random]
File View Debugger Tools Window Help
Class Library
Basis (14/80)
MaterialFlow
Resources
InfomationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
mu_pedido
Models
excel
random
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Edit Navigate Objects Icons View Tools Help
EventController
Duplo click no mu_Pedido = duplicate da Entity
random
Console
PT
16:55
16/05/2015


sequenciamento_v3.spp - Tecnomatix Plant Simulation 11 - [.Models.random]
File View Debugger Tools Window Help
Class Library
Basis (14/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
mu_pedido
Models
excel
random
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Edit Navigate Objects Icons View Tools Help
EventController
.MUs.mu_pedido
Navigate View Tools Help
Name: mu_pedido
Label:
Conveying direction: 0 (forward)
Attributes
Graphics
Product Statistics
User-defined Attributes
New...
Edit...
Delete
Name Value Type C. I.. 3D
codPedido 0 integer *
moore 0 integer *
tempoEntrega 0 integer *
tempoProc 0 integer *
OK
Cancel
Apply
random
Console
PT
16:56
16/05/2015


sequenciamento_v3.spp - Tecnomatix Plant Simulation 11 - [.Models.random]
File View Debugger Tools Window Help
Class Library
Basis (14/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
mu_pedido
Models
excel
random
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Edit Navigate Objects Icons View Tools Help
EventController
tf_pedidos
random
Console
CAP NUM SCRL
PT
16:51
16/05/2015


sequenciamento_v3.spp - Tecnomatix Plant Simulation 11 - [.Models.random.tf_pedidos]
File View Debugger Tools Window Help
Class Library
Basis (14/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
mu_pedido
Models
excel
random
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
File Edit Format Navigate View Tools Help
integer 1
integer 2
integer 3
string
Cod_Pedido
Tempo_Processamento
Tempo_Entrega
random
tf_pedidos
Console
Ready
PT
16:53
16/05/2015

1ª etapa

***Class Library* (*Toolbox* utilizada)**	**Descrição**	***Data Type***	***Name***	***Method***	**Função do Método**	***Componentes de dados***
Material Flow	*EventController*	*Time*	EventController	----------	----------	*Time*
Mobile Units	MUs	*Integer*	mu_pedido	----------	----------	***User-defined Attributes*** CodPedido ***moore*** tempoEntrega tempoProc
Information Flow	*Variable* (n=1)	*Integer*	var_nPedidos	----------	----------	Número de pedidos n = 100
Information Flow	*TableFile*	*Integer*	tf_pedidos	----------	----------	Cod_Pedido Tempo_Processamento Tempo_Entrega
Information Flow	***Method***	***Integer***	***Init***	***Init***	**Deletar dados da Tabela no início**	**Cod_Pedido Tempo_Processamento Tempo_Entrega**
Information Flow	***Method***	***Integer***	**mtd_cria_pedidos**	***User-Defined-Attribute ⇒ mtd_geraPedidos***	**Gerar dados de pedidos aleatoriamente**	**Cod_Pedido Tempo_Processamento Tempo_Entrega**
User Interface	*Button* OK	*Integer*	.Models.random.botao_1	mtd_geraPedidos	Gerar as Mus a partir dos dados de pedidos gerados aleatoriamente	Cod_Pedido Tempo_Processamento Tempo_Entrega

Toolbox utilizada

1) ⇒ *Information Flow* ⇒ n = 1

var_nPedidos=100

name = var_nPedidos,

Data Type = *Integer*

Value = 100

Limitador do numero de pedidos a ser gerado aleatoriamente e em seguida processado no modelo como um objeto representado pela entidade MU

2) ⇒ *User Interface* ⇒ *Button* OK ⇒ aciona o método gera pedidos

Control = self.**mtd_geraPedidos**

Attributes and methods = **mtd_geraPedidos**

Path = **.Models.random.botao_1.mtd_geraPedidos**

Pasta ⇒ *User-defined Attributes* = **mtd_geraPedidos**

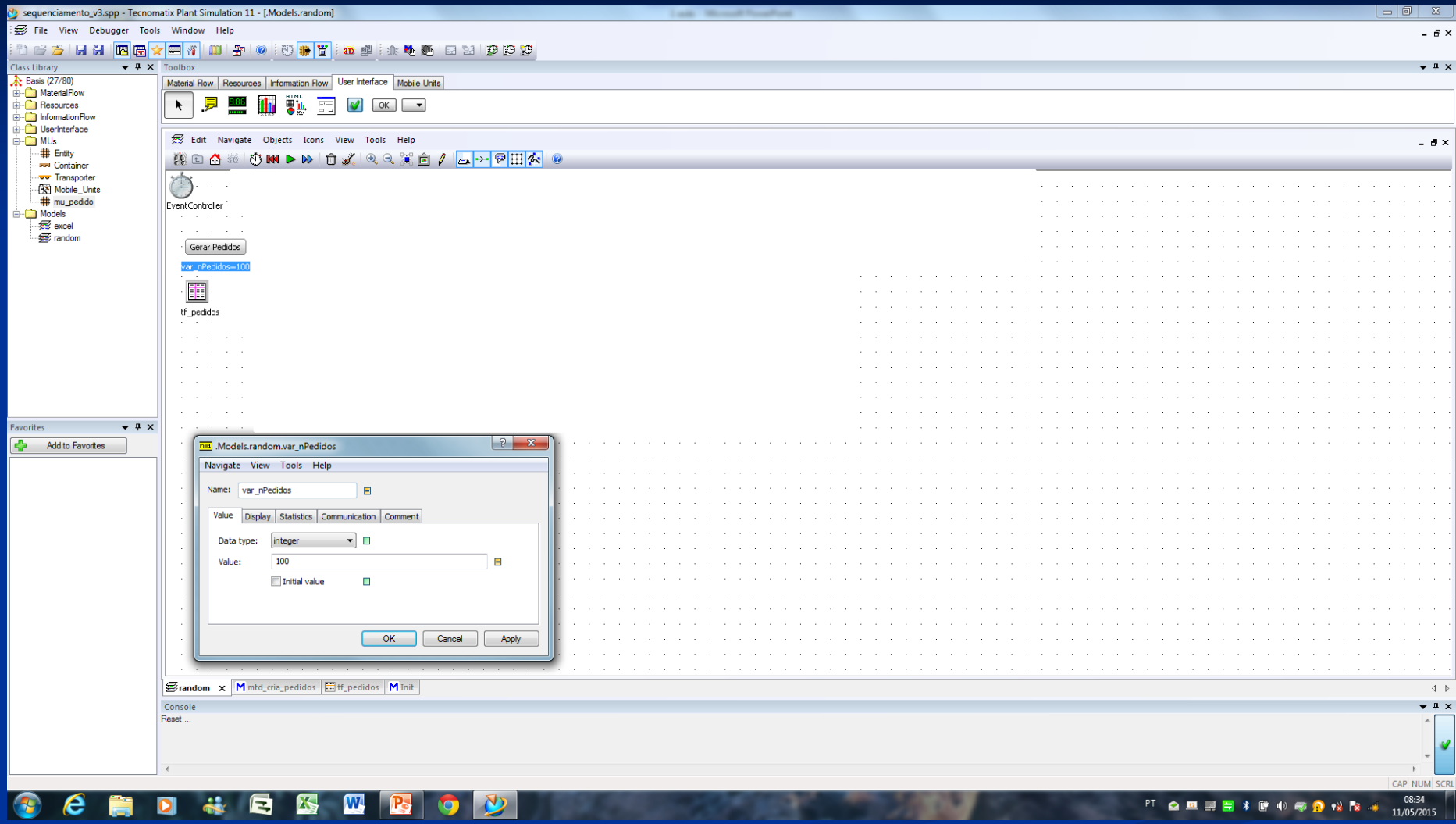
Identificação do botão no *Frame random*

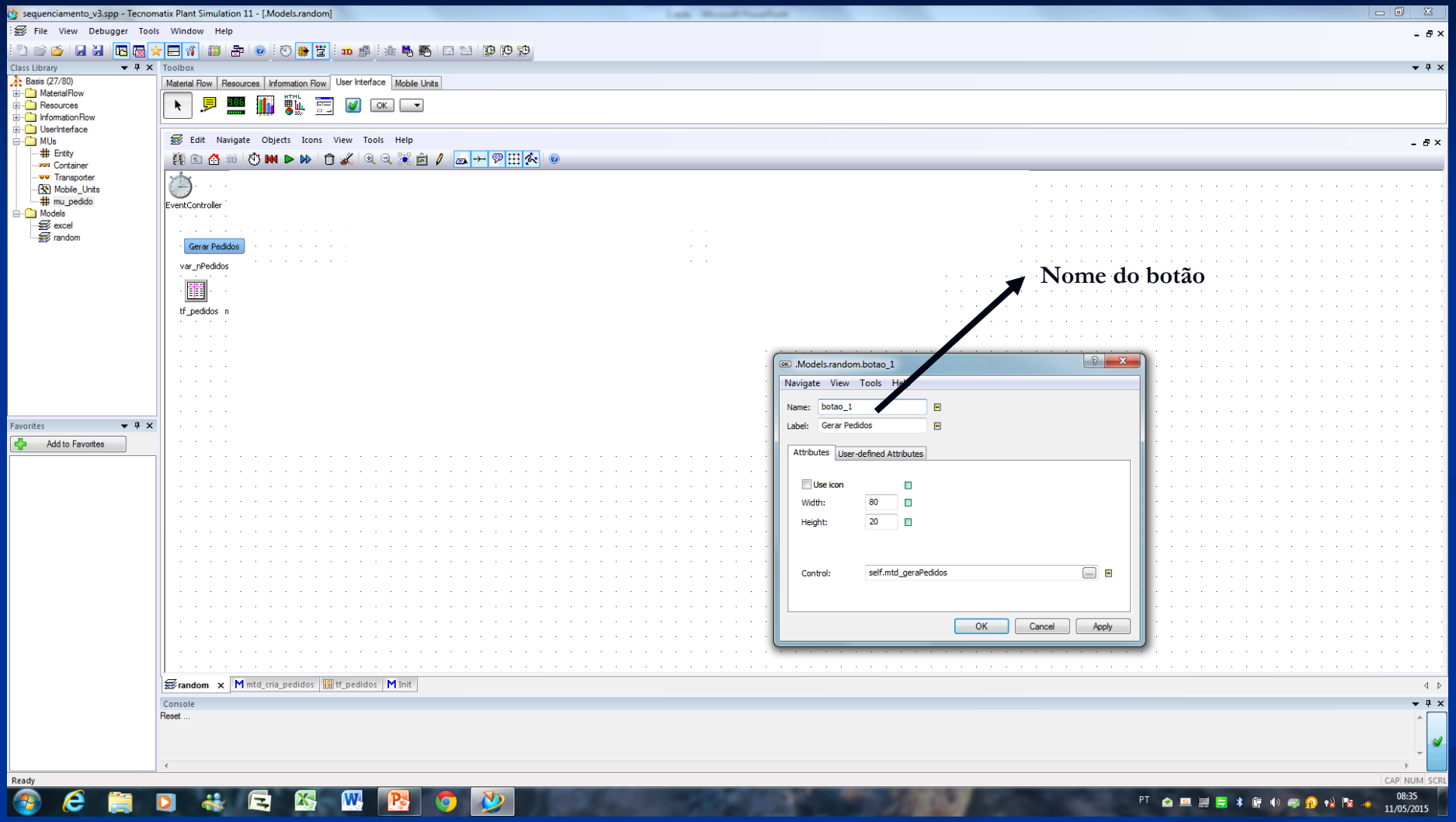
Identificação do método no botao_1 no *Frame random*

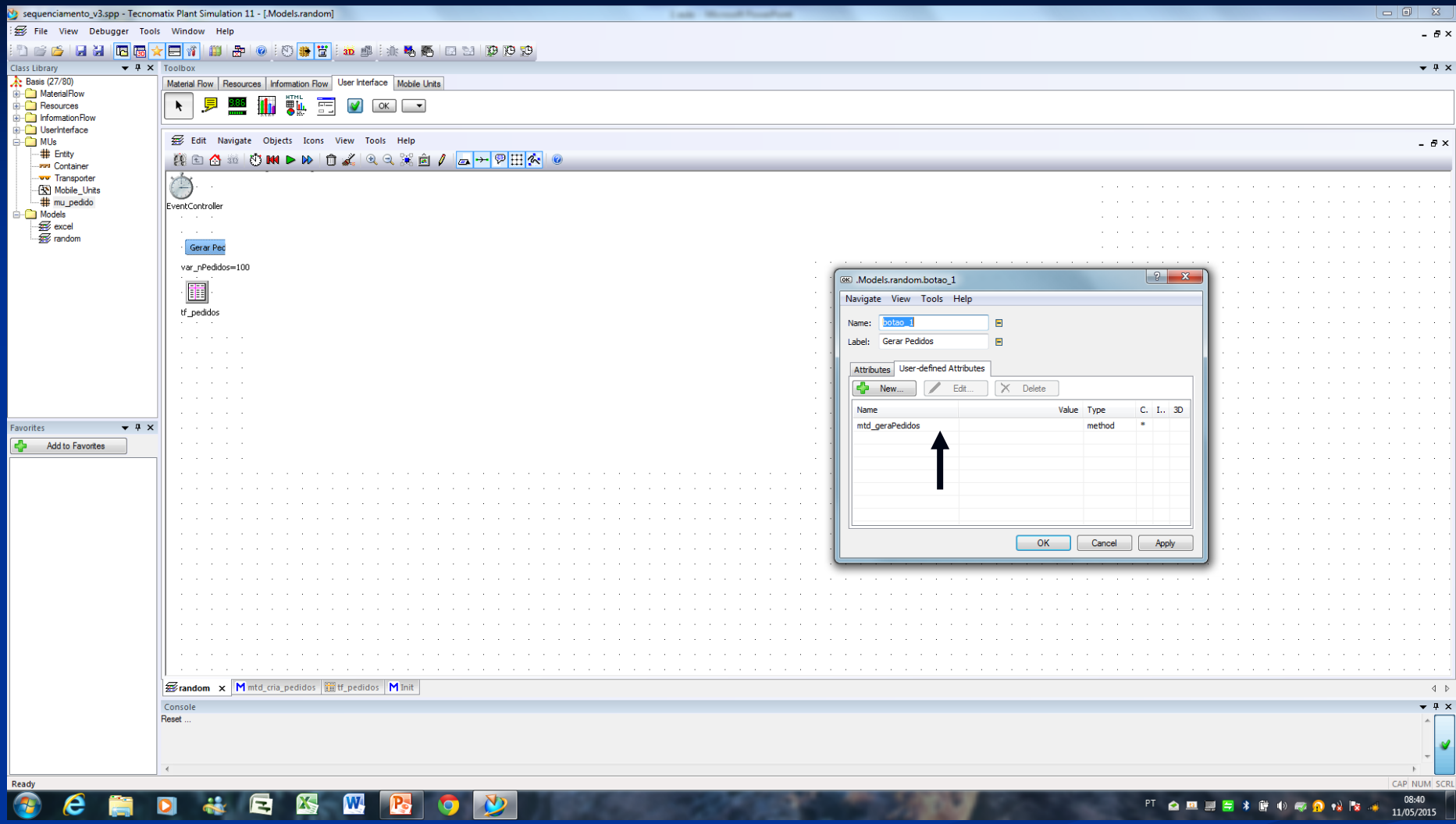
3) self.mtd_geraPedidos ⇒ .Models.random.botao_1

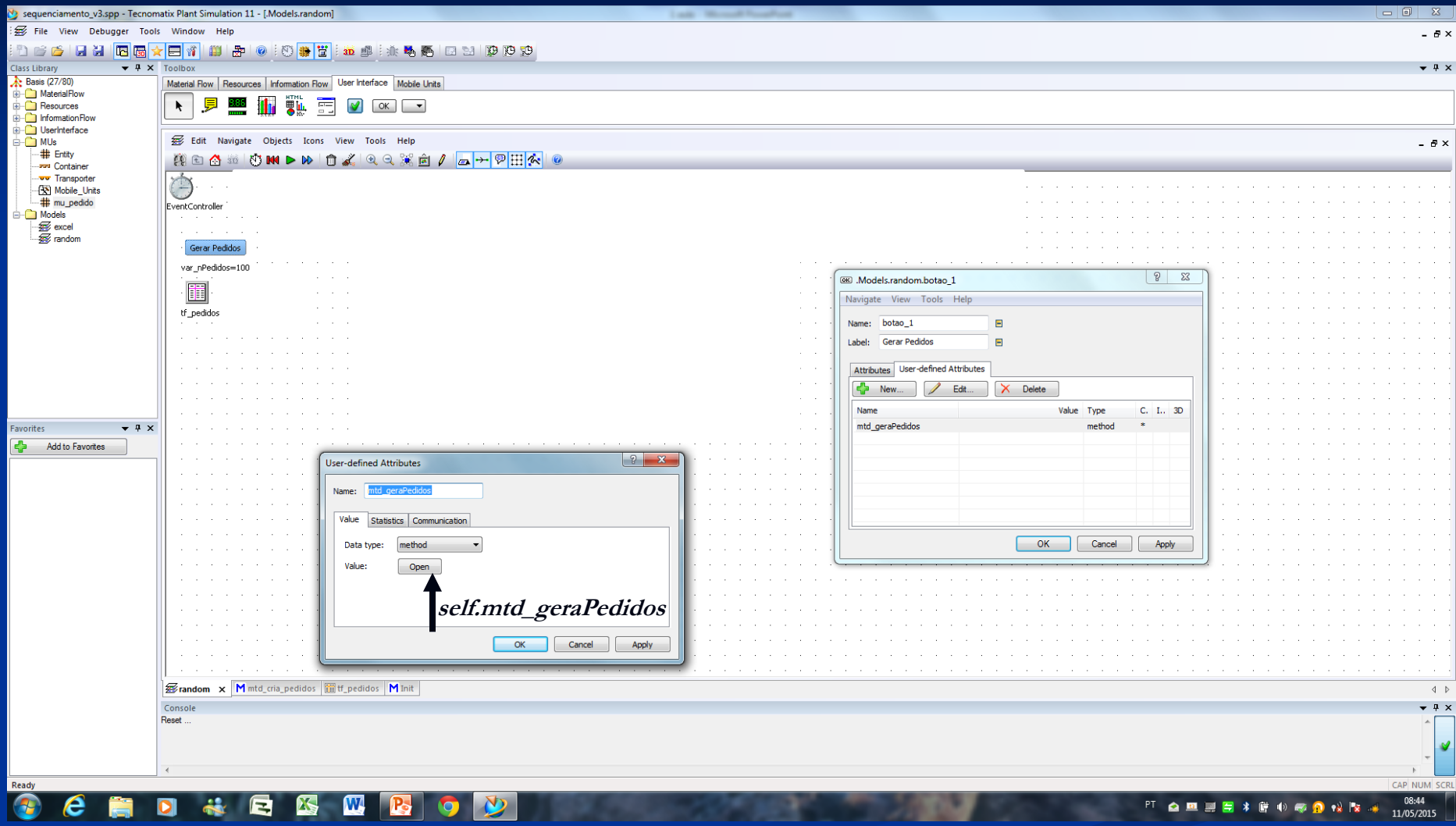
Self

O identificador anônimo *self* designa o método atualmente executado. Use o identificador *self* quando é provável que o método pode ser renomeado posteriormente. Caso contrário, você pode ter que mudar a cada instância do nome anterior para o novo nome.


sequenciamento_v3.spp - Tecnomatix Plant Simulation 11 - [.Models.random]
File View Debugger Tools Window Help
Class Library
Basis (27/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
mu_pedido
Models
excel
random
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
OK
Edit Navigate Objects Icons View Tools Help
EventController
Gerar Pedidos
var_nPedidos=100
tf_pedidos
.Models.random.var_nPedidos
Navigate View Tools Help
Name: var_nPedidos
Value
Display
Statistics
Communication
Comment
Data type: integer
Value: 100
Initial value
OK
Cancel
Apply
random
mtd_cria_pedidos
tf_pedidos
init
Console
Reset ...
CAP NUM SCRL
PT
08:34
11/05/2015


sequenciamento_v3.spp - Tecnomatix Plant Simulation 11 - [.Models.random]
File View Debugger Tools Window Help
Class Library
Basis (27/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
mu_pedido
Models
excel
random
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
OK
Edit Navigate Objects Icons View Tools Help
EventController
Gerar Pedidos
var_nPedidos
tf_pedidos
Nome do botão
.Models.random.botao_1
Navigate View Tools
Name: botao_1
Label: Gerar Pedidos
Attributes
User-defined Attributes
Use icon
Width: 80
Height: 20
Control: self.mtd_geraPedidos
OK
Cancel
Apply
random
mtd_cria_pedidos
tf_pedidos
Init
Console
Reset ...
Ready
PT
08:35
11/05/2015


sequenciamento_v3.spp - Tecnomatix Plant Simulation 11 - [.Models.random]
File View Debugger Tools Window Help
Class Library
Basis (27/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
mu_pedido
Models
excel
random
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
OK
Edit Navigate Objects Icons View Tools Help
EventController
Gerar Pec
var_nPedidos=100
tf_pedidos
.Models.random.botao_1
Navigate View Tools Help
Name: botao_1
Label: Gerar Pedidos
Attributes
User-defined Attributes
New...
Edit...
Delete
Name
Value
Type
C.
I..
3D
mtd_geraPedidos
method
OK
Cancel
Apply
random
mtd_cria_pedidos
tf_pedidos
Init
Console
Reset ...
Ready
PT
08:40
11/05/2015


sequenciamento_v3.spp - Tecnomatix Plant Simulation 11 - [.Models.random]
File View Debugger Tools Window Help
Class Library
Basis (27/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
mu_pedido
Models
excel
random
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
OK
Edit Navigate Objects Icons View Tools Help
EventController
Gerar Pedidos
var_nPedidos=100
tf_pedidos
User-defined Attributes
Name: mtd_geraPedidos
Value
Statistics
Communication
Data type: method
Value:
Open
self.mtd_geraPedidos
OK
Cancel
Apply
.Models.random.botao_1
Navigate View Tools Help
Name: botao_1
Label: Gerar Pedidos
Attributes
User-defined Attributes
New...
Edit...
Delete
Name
Value
Type
C.
I..
3D
mtd_geraPedidos
method
OK
Cancel
Apply
random
mtd_cria_pedidos
tf_pedidos
Init
Console
Reset ...
Ready
PT
08:44
11/05/2015

Variáveis do pedido

1) Código do Pedido
2) Tempo de Processamento
3) Tempo de Entrega – *Due Date*

Variável de entrada (*Input*)	***Table* Conotação Inicial *Input – SimTalk***	**Método**		**Atribuição *SimTalk* no modelo**
		Identificação do método	**Descrição**	
Código do Pedido	cod_Pedido (**1ª Coluna - *Table***)	mtd_cria_pedidos (2º método)	pedido.codPedido := tf_pedidos["Cod_Pedido",i];	atribui a MU o Código do Pedido
Tempo de processamento	Tempo_Processamento (**2ª Coluna - *Table***)	mtd_cria_pedidos (2º método)	**pedido.tempoProc** := tf_pedidos["Tempo_Processamento",i];	atribui a MU o tempo de processamento
Tempo de Entrega	Tempo_Entrega (**3ª Coluna - *Table***)	mtd_cria_pedidos (2º método)	***pedido.tempoEntrega*** := tf_pedidos["Tempo_Entrega",i];	atribui a MU o tempo de entrega

1º Método

Deletar dados das Tabelas para dar início a novo processo de simulação

INIT

Processo:

1) Acionar Botão para acionar o método de gerar pedidos;
2) Acionar *Play* para acionar método INIT para limpar Tabelas e dar início a partir dos demais métodos da simulação dos processos.

Método *Init*

```
--Metodo executado no inicio da simulacao (sempre que se clica em "play")
is
do
        --Limpa as tabelas de resultados
        tf_resultados_FIFO.delete();
        tf_resultados_SPT.delete();
        tf_resultados_EDD.delete();
        tf_resultados_MOORE.delete();
                --Limpa a tabela de comparacao
        tf_comparacao.delete();

        --Aciona o metodo para criar os pedidos ⇒ mtd_cria_pedidos;

end;
```

1ª Etapa

Input do Modelo

2º Método

Gera Pedidos Aleatórios

self.mtd_geraPedidos

2º Método

SELF.MTD_GERAPEDIDOS

Variável – i – inteiro – linha da Tabela
Fazer (*do*):
1) **Deletar dados da tabela tf_pedidos**
2) ***Loop* – para i = 1 até o limite definido na variável ⇒ var_nPedidos (número máximo de pedidos igual a 100) – *loop* –**
tf_pedidos ⇒ cabeçalho da Tabela Pedidos – 1 coluna – i – número sequencial dos pedidos a partir de i = 1
2 coluna – z_uniform(1,1,5) – gera números aleatórios a partir de uma distribuição uniforme entre entre 1 e 5 a partir da semente 1
3 coluna – z_uniform(1,*tf_pedidos[2,1]*,90) – gera números aleatórios a partir de uma distribuição uniforme entre a *Due Date* do pedido a 90 para evitar que ocorra tarefas com atraso.
Tarefas com atraso não se aplica o procedimento de Moore.

```
is
    i : integer;
do
    tf_pedidos.delete;
    for i := 1 to var_nPedidos loop
        tf_pedidos[1,i] := i;
        tf_pedidos[2,i] := z_uniform(1,1,5);
        tf_pedidos[3,i] := z_uniform(1,tf_pedidos[2,i],90);
    next;
end;
```

Conceito

Random Number Stream

Fundamentação teórica

Conceitos

Random number stream

Random Number Seed Values

Syntax - setSeedTable(SeedTable:<table>);

A função ***setSeedTable sets the Random Number Seed Values*** para a criação dos ***random number streams***.

* Os ***seed values*** somente se aplica as funções de distribuição, tal como ***z_uniform, z_normal***, etc.

Cada um dos objetos do fluxo de materiais automaticamente usa um ***random number stream*** próprio. Você pode considerar o ***random number stream*** com o ***attribute RandomSeed***.

* O *Plant Simulation* aplica novos números aleatórios imediatamente e reestabelece (*resets*) todos os ***random number streams*** com os ***new seed values***.

Random number stream

Parâmetro

O parâmetro ***SeedTable*** do tipo de dados da Tabela designa uma Tabela com uma ou duas colunas ou uma lista.

- A primeira coluna contem o ***random number seed values*** (inteiro).
- A segunda coluna contem os comentários dos valores ***random number seed*** (***string***). O número do conjunto de linhas do ***random number stream***.

Exemplo:

```
is
t : table[integer];
do
    t.create;
    t[1,6] := 1;
    setSeedTable(t);
end;
```

⇒ **Criar a Tabela t (declarada como t : table[integer]) ⇒ 1 coluna e 6 linhas ⇒ atribuir para todas as células o valor igual a 1.**

Random number stream

Exemplo:

```
is
t : table[integer];
do
    t.create;
    t[1,6] := 1;
    setSeedTable(t);
end;
    Example:  is
    t : table[integer];
do
    t.create;
    t[1,6] := 1;
    t[2,6] := "Explain usage";
    setSeedTable(t);
end;
```

Random number stream

- O comando do *menu*: *Random Number Seed Values* abre os valores semente da Tabela. Contêm os valores da semente da *random number streams*, a qual se aplica apenas as *Distribution Functions*, tal como z_uniform, z_normal, etc.
- Por exemplo, z_uniform(3,1,10), gera uniformemente números distribuídos entre 1 e 10, usando o *random number stream* 3. Você pode adicionar mais *random number streams* ou editar os *streams* na Tabela.

3º Método

mtd_cria_pedidos

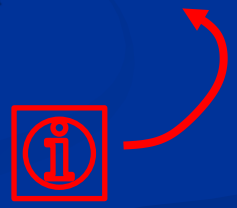
Resumo – Variáveis do pedido

Variável de entrada (*Input*)	**Conotação Inicial *Input – SimTalk***	**Método**		**Atribuição *SimTalk* no modelo**
		Identificação do método	**Descrição**	
Código do Pedido	cod_Pedido (**1ª Coluna - *Table***)	mtd_cria_pedidos (3º método)	pedido.codPedido := tf_pedidos["Cod_Pedido",i];	atribui a MU o tempo de processamento
Tempo de processamento	Tempo_Processamento (**2ª Coluna - *Table***)	mtd_cria_pedidos (3º método)	**pedido.tempoProc** := tf_pedidos["Tempo_Processamento",i];	atribui a MU o tempo de processamento
Tempo de Entrega	Tempo_Entrega (**3ª Coluna - *Table***)	mtd_cria_pedidos (3º método)	***pedido.tempoEntrega*** := tf_pedidos["Tempo_Entrega",i];	atribui a MU o tempo de processamento

mtd_cria_pedidos

```
--Metodo faz a leitura dos pedidos na tabela tf_pedidos e cria as MUs correspondentes
--em cada um dos objetos "sorter"
is
        pedido : object;
do
     for local i := 1 to tf_pedidos.YDim loop                    --roda toda a lista de pedidos
       pedido := .MUs.mu_pedido.create(srt_FIFO);                --cria no primeiro sorter uma MU do tipo "mu_pedido"
       pedido.codPedido := tf_pedidos["Cod_Pedido",i]; --atribui a MU criada um codigo igual ao da tabela de pedidos
       pedido.tempoProc := tf_pedidos["Tempo_Processamento",i];       --atribui a MU o tempo de processamento
       pedido.tempoEntrega := tf_pedidos["Tempo_Entrega",i];   --atribui a data de entrega

       --Copia para os demais sorters a MU criada no sorter srt_FIFO
       pedido.create(srt_SPT);
       pedido.create(srt_EDD);
       pedido.create(srt_MOORE);
next;

       --aciona o metodo de ordenacao de cada um dos sorters
        srt_FIFO.sort;
        srt_SPT.sort;
        srt_EDD.sort;
        --para o srt_moore, aciona o metodo personalizado criado para representar o algoritimo
         srt_MOORE.mtd_moore;
end;
```

Declarar pedido como objeto ⇒ cria as MUs



Conceito

Variável local

Local Variable

Declarar uma variável em algum lugar dentro do código fonte

Você pode também declarar uma variável local em algum lugar dentro do código fonte. Declare sua variável com a ***keyword*** ⇒ **local**, seguida de um ou alguns identificadores para as variáveis locais que você está definindo. Na sequência você pode inserir uma coluna e definir o tipo de dado da variável local.

⇒ **Exemplo: local x, y : integer**;

Uma variável local, a qual você declarou é visivel a partir da linha, na qual você a identificou, no final do código fonte. Exatamente como uma variável local, na qual você declarou entre **is e do**, uma variável local definida via local é inicializada com um valor inicial. Comparar Compare **Data Types** em **Local Variables**.

Você pode também atribuir um valor quando você declarar a variável local.

⇒**Example: is**

```
do
   local a : length := Track1.length;
   local b : length := Track2.length;
   local c : length := sqrt(a*a + b*b); ⇒ raiz quadrada de (a*a + b*b)
   print c;
end;
```

Local Variable

Quando você atribuir um valor para a variável local enquanto você declarar a mesma, a determinação ou a declaração do **data type** é opcional. O **data type** da variável é então determinada pelo valor atribuído.

⇒ **Example: is**

```
do
   local a := Track1.length;
   local b := Track2.length;
   local c := sqrt(a*a + b*b);
   print c;
end;
```

No exemplo acima as variáveis **a e b** devem assumir o **data type** comprimento. A variável c devem assumir o **data type** real, assim como a função ***sqrt*** retorna um valor do ***data type real***.

Local Variable

For-Loop como uma ***Local Loop Variable***

Como um caso especial você pode usar a ***keyword*** local em um ***for-loop*** para definir a ***loop variable***. A ***loop variable*** torna-se então somente visível dentro do ***loop***. For a do ***loop*** o identificador ao contrário aponta para um objeto com este nome se este objeto exite, por exemplo para uma variável global.

⇒ Example: is

```
do
   -- the loop variable 'i' is only visible inside the loop
   for local i := 1 to ParallelProc.numMU loop
      print ParallelProc.mu(i);
   next;
   print i; -- the loop variable is not visible here, so 'i' must be the name of an object
   -- another loop with its own loop variable 'i'
   for local i := ParallelProc.numMU downto 1 loop
      print ParallelProc.mu(i);
   next;
   -- the local variable 'i' is visible from here to the end of the source code
   local i := 1.234;
   print i; -- prints 1.234
   for local i := 1 to 10 loop -- does not compile, because 'i' is already declared
   next;
end;
```

downto
The keyword downto is part of The for-loop, together with the keywords for, next, and to.
Para a variável local i := (atribuir)

MU

numMu

Syntax: <MU_path>.numMu;

O método ***numMu*** retorna o número da MUs localizado no espaço de carregamento da MU designado por ***<MU_path>***.

Return Value

A valor de retorno tem o ***data type*** inteiro.

⇒ ***Example: print .MUs.transporter:1.numMu***;

MU

mu

Syntax: <Path>.mu[(Index:<integer>)];

O método ***mu*** acessa todas as MUs cujo ponto de reserva está localizado no objeto designado por<Path>.

Parameter

Quando você insere o índice do tipo de dado opcional ***(optional parameter Index)*** do ***data type –*** inteiro, o *Plant Simulation* acessa a MU designada pelo parâmetro do ***data type integer***. Para um parâmetro maior do que o número de peças reservadas no objeto, o *Plant Simulation* retorna ***VOID***. Quando você não insere o parâmetro opcional, o *Plant Simulation* acessa a primeira MU.

* Para o objeto ***Buffer Plant Simulation*** acessa as MUs no ***Buffer*** na ordem em que eles entram no ***buffer***, independente do ***Buffer type*** (**sobreposição ou fila - *Stack or Queue***) que você selecionou. O método ***mu*** não refere ao local de ***storage place*** da ***MU*** no ***Buffer***, mas é simplesmente um contador para as ***MUs*** localizadas no ***Buffer***. Para acessar a próxima MU, no qual deve existir o ***Buffer*** de acordo com o ***buffer type Stack or Queue***, usado no método cont.

Return Value

Retorna o valor do tipo de dado do objeto.

⇒ Example: is

```
i : integer;
do -- accesses all MUs
   for i := 1 to ParallelProc.numMu loop
        print ParallelProc.mu(i).name;
        next;
end;
```

Método Cont.

cont

Syntax: <Path>.cont;

O método ***cont*** retorna a MU localizada no objeto designado pelo ***<Path>*** que tem uma capacidade de 1. Ele retorna ***VOID*** se nenhuma MU encontra-se localizada no objeto.

* Quando o ***Worker***, que está carregando uma peça, descarrega aquela peça, o método ***Worker.cont*** retorna a peça descarregada.

Return Value

O valor retornado tem o tipo de dado do objeto (***data type object***).

⇒ Exemplo: if MySingleProc.cont /= VOID then

```
    print MySingleProc.cont.name;
end;
parallelProc.cont.CurrIconNo := 1;
```

MU

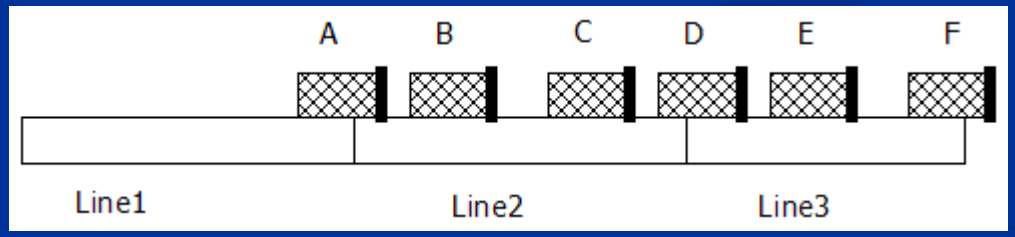
muPart

O método ***muPart*** acessa todas as MUs localizadas no objeto designado por <Path> como um todo ou em parte.

Parâmetro

Quando você insere o ***optional parameter Index*** do ***data type integer***, o ***Plant Simulation*** acessa a MU designada pelo parâmetro do ***data type integer***. Quando você não insere o parâmetro, o ***Plant Simulation*** acessa a primeira ***muPart*** disponível.

⇒ exemplo: as MUs A, B, C, D, E e F estão localizadas no comprimento ***length-oriented objects Line1, Line2 e Line3***.



Os pontos reservados de todas as MUs estão localizados na sua extremidade direita. Os valores para as ***muPart*** estão listados na Tabela do próximo *slide*.

MU

Call	*Returns*
line1.MUPart(1)	MU A
line1.MUPart(2)	VOID
line2.MUPart(1)	MU D
line2.MUPart(2)	MU C
line2.MUPart(3)	MU B
line2.MUPart(4)	MU A
line2.MUPart(5)	VOID

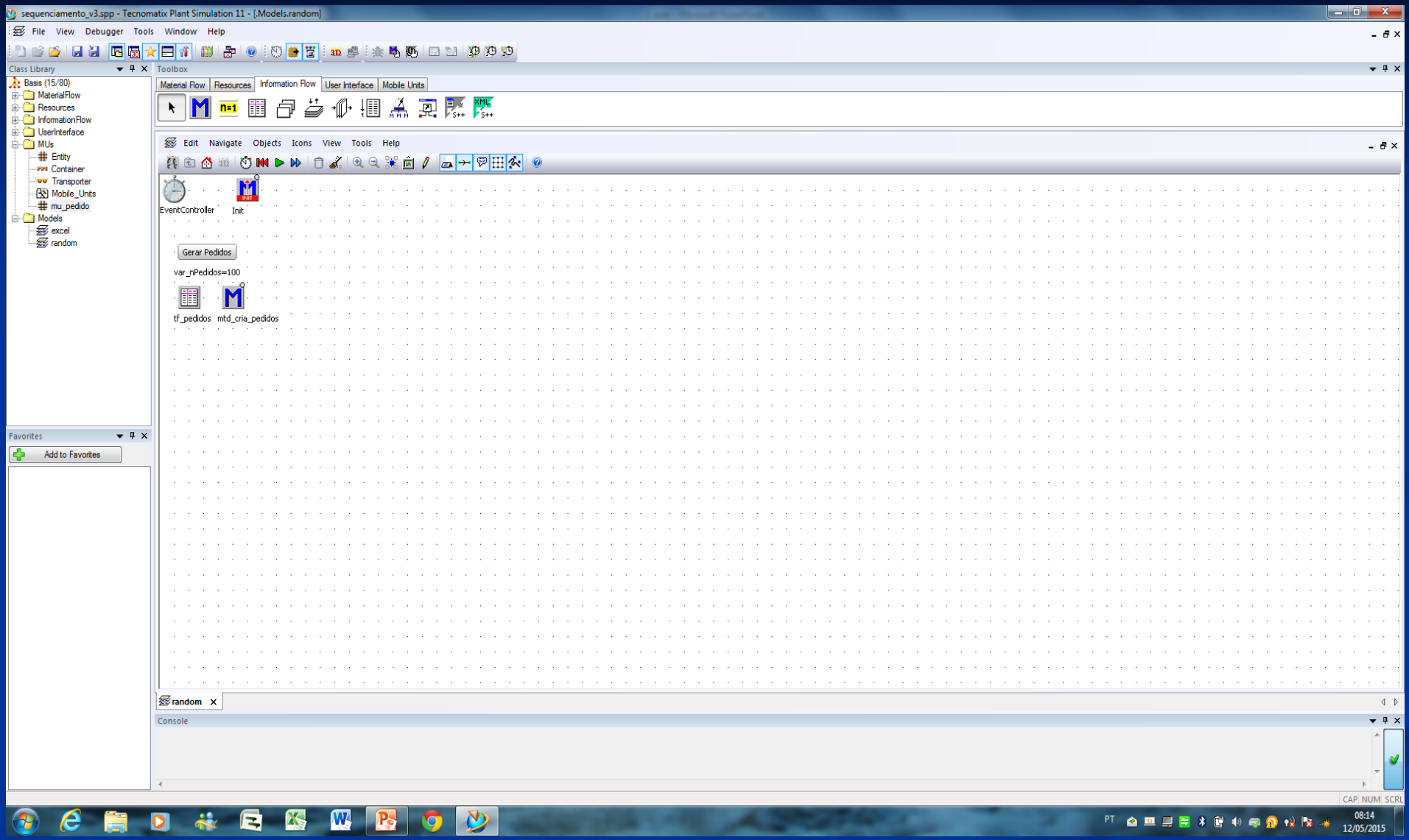
Return Value

O valor de retorno tem o tipo de dado do objeto.

Se nenhum ***muPart*** é localizado no objeto ou o ***muPart*** chamado não existe, o método retorna vazio (***VOID***).

⇒ ***Example: line1.muPart(1); -- returns VOID.***

line2.muPart(1); -- returns the mu C


sequenciamento_v3.spp - Tecnomatix Plant Simulation 11 - [.Models.random]
File View Debugger Tools Window Help
Class Library
Basis (15/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Entity
Container
Transporter
Mobile_Units
mu_pedido
Models
excel
random
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Edit Navigate Objects Icons View Tools Help
EventController
Init
Gerar Pedidos
var_nPedidos=100
tf_pedidos
mtd_cria_pedidos
random
Console
PT
08:14
12/05/2015

2ª Etapa

Procedimento de Moore

Resumo – 2ª etapa

***Class Library* (*Toolbox* utilizada)**	**Descrição**	***Data Type***	***Name***	***Method***	**Função do Método**	**Variáveis**
Material Flow	*Sorter*	*Integer*	st_MOORE	***Attributes – moore*** mtd_moore ***user-Defined-Attribute ⇒ Moore***	Procedimento de Moore	Tempo_Processamento Tempo_Entrega
Material Flow	*SingleProc*	*Integer*	sp_MOORE	***user-Defined-Attribute ⇒ OnEntrance***	Resultados	Soma tempo total de atraso Maior tempo de atraso Número de tarefas atrasadas
Material Flow	*Drain*	*Integer*	dm_MOORE	***user-Defined-Attribute ⇒ OnEntrance***	Resultados	Soma tempo total de atraso Maior tempo de atraso Número de tarefas atrasadas

4º Método

mtd_moore

Definição das variáveis do modelo

```
is
   sequencia          : table;    --salva a sequencia de ordenamento
   i,j                : integer; --apenas indices para percorrer as tabelas
   nTotal             : integer; --salva o numero total de trabalhos a serem ordenados
   nInicio            : integer; --salva o numero de trabalhos que compoe a "sequencia atual"
   aux                : object;  --variavel auxiliar para efetuar trocas simples
   somaDeTempos       : integer; --salva a soma de tempos da sequencia atual
   nAtrasado          : integer; --salva a posicao do primeiro trabalho atrasado na "sequencia atual"
   maiorProcTime      : integer; --salva o maior tempo de processamento na ssequencia de confronto (subsequencia da sequencia atual)
   nMaiorProcTime : integer; --salva a posicao do tempo de maior processamento na sequencia de confronto (nMaiorProcTime <= nAtrasado)
```

1) := ⇒ **atribui**

2) = ⇒ **compara**

3) : ⇒ **declaração de variável (*array*) – ver anexo III**

Do

```
do
  --aloca memoria para a tabela
  sequencia.create;

  srt_MOORE.contentsList(sequencia); --cria a sequencia inicial com base nas MUs contidas no objeto sorter Moore (srt_MOORE)
  nTotal := sequencia.YDIM;          --salva o numero total de tarefas encontradas

  --valores iniciais
  nInicio                := nTotal;
  somaDeTempos           := 0;
  maiorProcTime          := 0;
  nAtrasado              := 1; --qualquer valor diferente de 0

  --bubble sort (ordenando pela menor data de entrega)
  for i:=1 to nTotal loop
        for j:=1 to nTotal-1 loop
        if sequencia[1,j].tempoEntrega > sequencia[1,j+1].tempoEntrega then
                              aux := sequencia[1,j];
                              sequencia[1,j] := sequencia[1,j+1];
                              sequencia[1,j+1] := aux;
                        end;
                next;
    next;
```

Do

```
do
   while (nAtrasado /= 0 and nInicio >0) loop  --executa enquanto houverem trabalhos atrasados na sequencia atual

         --encontra o 1 atrasado para definir a subsequencia de confronto
          nAtrasado := 0;
          for i:=1 to nInicio loop                                    --percorre toda a sequencia atual
                somaDeTempos := somaDeTempos + sequencia[1,i].tempoProc; --soma os tempos de processamento
                if(somaDeTempos > sequencia[1,i].tempoEntrega) then          --verifica se o trabalho ira atrasar
                      nAtrasado := i;                               --salva a posicao do trabalho atrasado
                      exitloop;                                           --para de percorrer a lista caso ja tenha encontrado trabalho atrasado
                end;
       next;


          if(nAtrasado /= 0) then --caso tenha encontrado uma tarefa atrasada..

                      --encontra o maior procTime da subsequencia de confronto
                      for i := 1 to nAtrasado loop                  --percorre toda a sequencia de confronto
                            if sequencia[1,i].tempoProc > maiorProcTime then   --procura pelo maior tempo de processamento
                                  maiorProcTime := sequencia[1,i].tempoProc;
                                  nMaiorProcTime := i;                  --salva o indice do maior tempo de processamento
                             end;
                     next;
```

```
do
    --desloca o maior procTime para o final da sequencia  e rearranja a mesma
    aux := sequencia[1,nMaiorProcTime];
    for i:=nMaiorProcTime to nTotal-1 loop
          sequencia[1,i] := sequencia[1,i+1];
     next;
     sequencia[1,nTotal] := aux;

      --reduz o tamanho da lista
      nInicio := nInicio -1;
      --reseta a soma de tempos
      somaDeTempos := 0;
      --reseta o maior procTime
       maiorProcTime := 0;
    end;
end;
     --define o atributo em cada uma das MUs informando qual a sua posicao na sequencia encontrada
      for i:=1 to nTotal loop
              sequencia[1,i].moore := i;
       next;
              --faz o objeto sorter (str_MOORE) ordenar as MUs conforme a sequencia encontrada
         srt_MOORE.sort;
end;
```

5º Método

OnEntrance

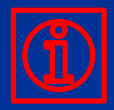
st_MOORE

Do

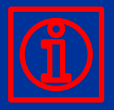
```
is
do
        ?.procTime := @.tempoProc;
end;
```

O tempo de processamento da MU que está sendo processada

Qual o tempo de processamento utilizar?
Identificador anônimo



Mtd_cria_pedido
pedido.tempoProc := tf_pedidos["Tempo_Processamento",i];--atribui a MU o tempo de processamento



Definido no segundo método – o tempo de processamento do pedido é definido por tempoProc

6º Método

OnEntrance

sp_MOORE

Do

```
is
    linha : integer;
do
    linha := tf_resultados_MOORE.YDim +1;
    tf_resultados_MOORE[1,linha] := @.codPedido;
    tf_resultados_MOORE[2,linha] := @.tempoEntrega;
    tf_resultados_MOORE[3,linha] := (eventController.SimTime);

    if(tf_resultados_MOORE[3,linha] < tf_resultados_MOORE[2,linha]) then
          tf_resultados_MOORE[4,linha] := 0;
          else
             tf_resultados_MOORE[4,linha] := tf_resultados_MOORE[3,linha] - tf_resultados_MOORE[2,linha]
        end;
end;
```

Última etapa

Apontamento dos Resultados e
Comparação entre os métodos

7º Método

Compara os resultados

endSim

Método endsim

```
--Metodo executado quando a simulacao chega ao fim
--copia os resultados de cada um dos metodos e preenche a tabela "tf_comparacao"
is
        i :integer;     --indice para percorrer todas as linhas da tabela
        maxT : integer; --variavel para salvar o maior atraso encontrado
        nT : integer;   --variavel para o numero de atrasos encontrados
        sumT : integer; --variavel para somar todos os tempos de atraso

do

-----------------------------------------------------------
    for i:=1 to tf_resultados_FIFO.YDIM loop          --percorre toda a tabela de resultados do metodo FIFO
        if(tf_resultados_FIFO[4,i] > 0) then          --caso o tempo de atraso seja maior do que zero..
            nT := nT +1;                               --incrementa o numero de trabalhos atrasados
            sumT := sumT + tf_resultados_FIFO[4,i];    --soma o atraso ao tempo total de atrasos
            maxT := max(maxT, tf_resultados_FIFO[4,i]); --verifica se o atraso encontrado e o maior atraso

            end;

next;
```

Método endsim

```
--preenche os dados na tabela tf_comparacao
tf_comparacao["SUM(T)","FIFO"] := sumT;
tf_comparacao["Max(T)","FIFO"]  := maxT;
tf_comparacao["n(T)","FIFO"]       := nT;
--reseta as variaveis
sumT     := 0;
maxT     := 0;
nT                  := 0;
-------------------------------------------------------------
for i:=1 to tf_resultados_SPT.YDIM loop
        if(tf_resultados_SPT[4,i] > 0) then
                nT := nT +1;
                sumT := sumT + tf_resultados_SPT[4,i];
                maxT := max(maxT, tf_resultados_SPT[4,i]);
        end;
next;
```

Método endsim

```
tf_comparacao["SUM(T)","SPT"]   := sumT;
tf_comparacao["Max(T)","SPT"]   := maxT;
tf_comparacao["n(T)","SPT"]     := nT;
sumT    := 0;
maxT    := 0;
nT              := 0;
-------------------------------------------------------------
for i:=1 to tf_resultados_EDD.YDIM loop
        if(tf_resultados_EDD[4,i] > 0) then
                nT := nT +1;
                sumT := sumT + tf_resultados_EDD[4,i];
                maxT := max(maxT, tf_resultados_EDD[4,i]);
        end;
next;
tf_comparacao["SUM(T)","EDD"]  := sumT;
tf_comparacao["Max(T)","EDD"]   := maxT;
tf_comparacao["n(T)","EDD"]     := nT;
```

Método endsim

```
sumT     := 0;
maxT     := 0;
nT                  := 0;
-------------------------------------------------------------
for i:=1 to tf_resultados_MOORE.YDIM loop
        if(tf_resultados_MOORE[4,i] > 0) then
                nT := nT +1;
                sumT := sumT + tf_resultados_MOORE[4,i];
                maxT := max(maxT, tf_resultados_MOORE[4,i]);
        end;
next;
tf_comparacao["SUM(T)","MOORE"] := sumT;
tf_comparacao["Max(T)","MOORE"] := maxT;
tf_comparacao["n(T)","MOORE"]  := nT;
-------------------------------------------------------------
end;
```

ANEXO I

ENTITY & FRAMING

MUs

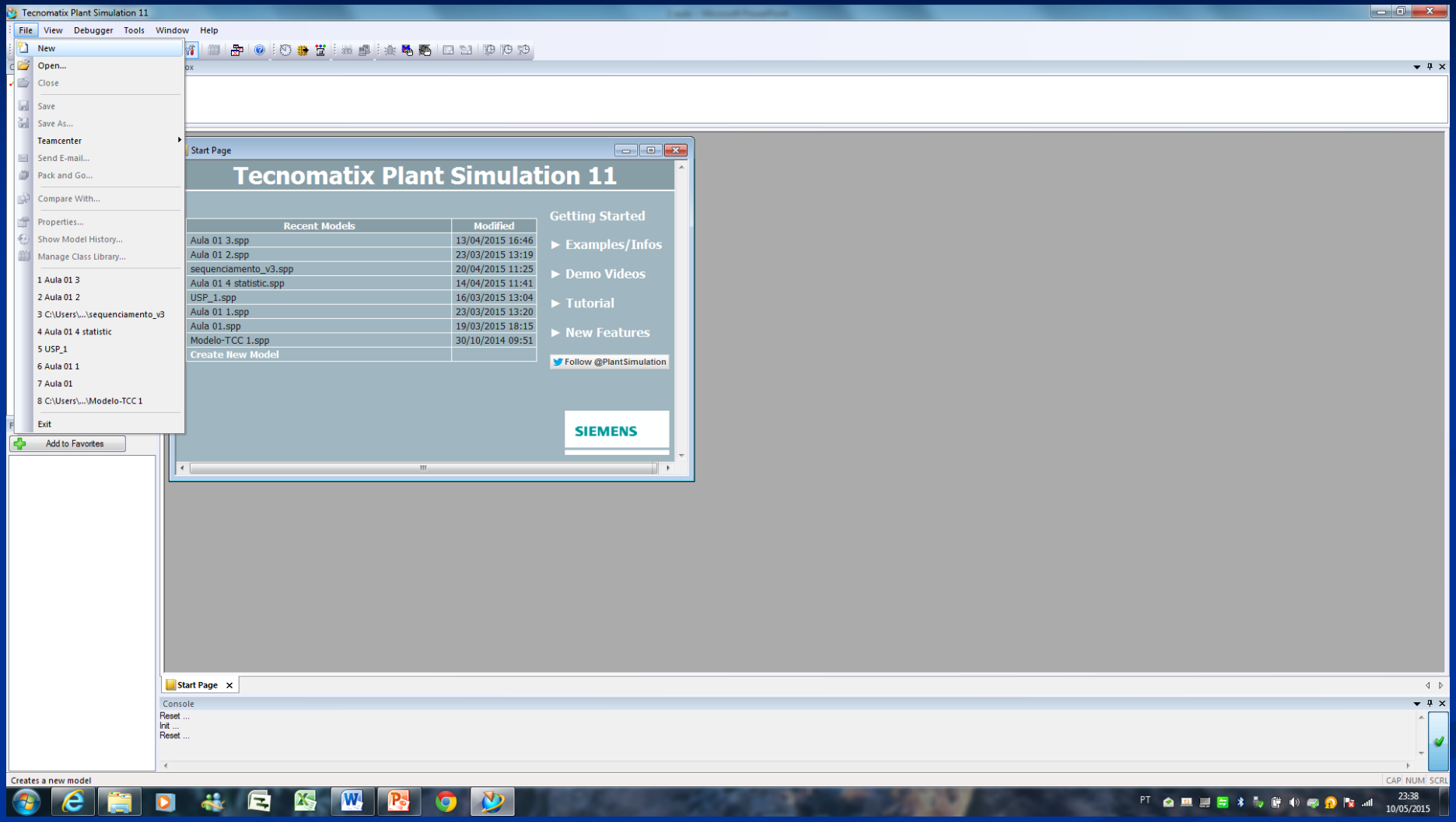
ENTITY

Entity - MUs

- A *Entity* é um objeto que está em movimento no fluxo de materiais (*moving material flow object*) sem capacidade de carregamento que se move através da planta no próprio fluxo de materiais do objeto (*without loading capacity that moves through a plant on the material flow objects proper*). A *Entity* representa peças sendo produzidas e transportadas, mas não transportando outras peças em produção.

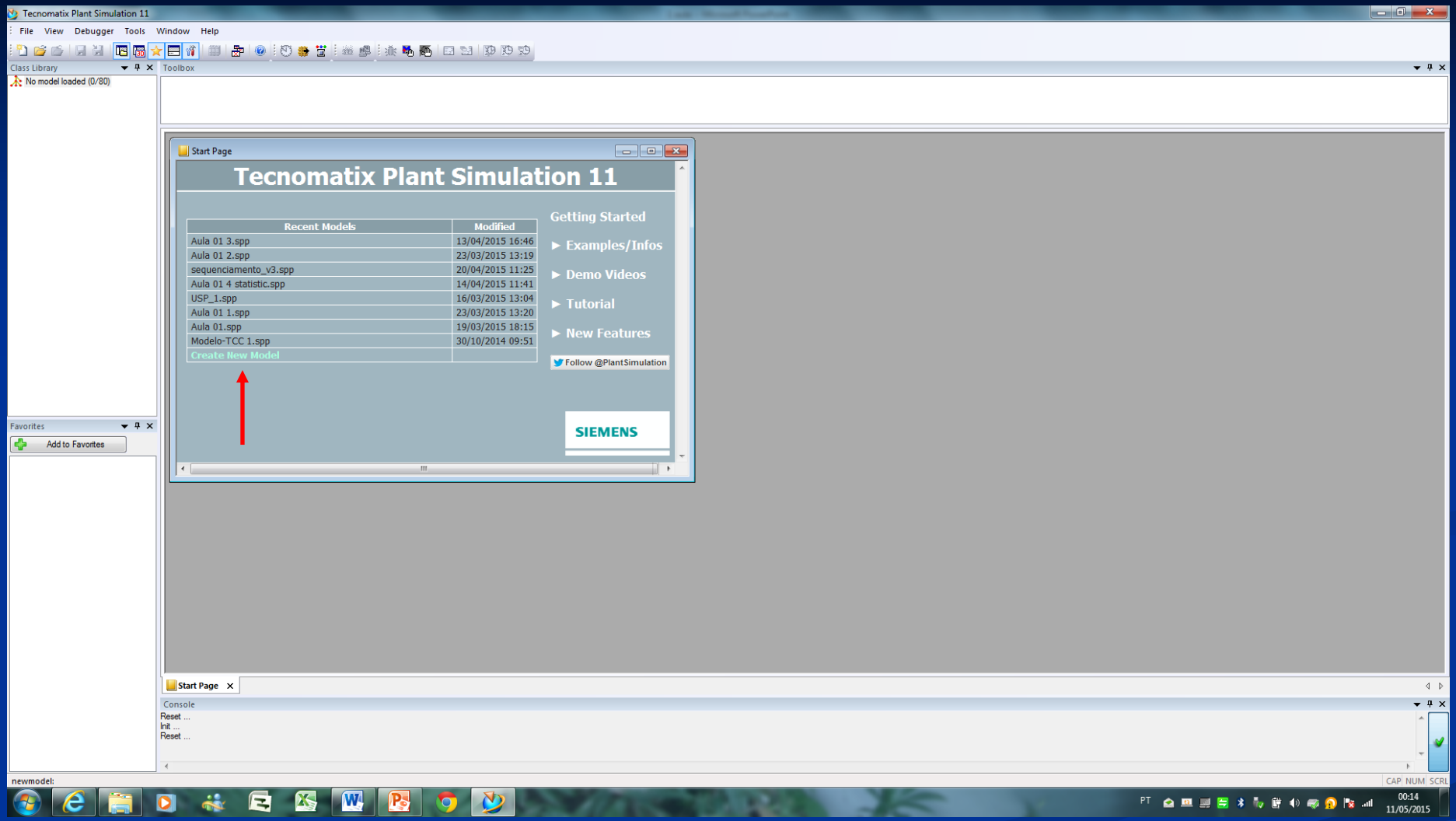
Frame

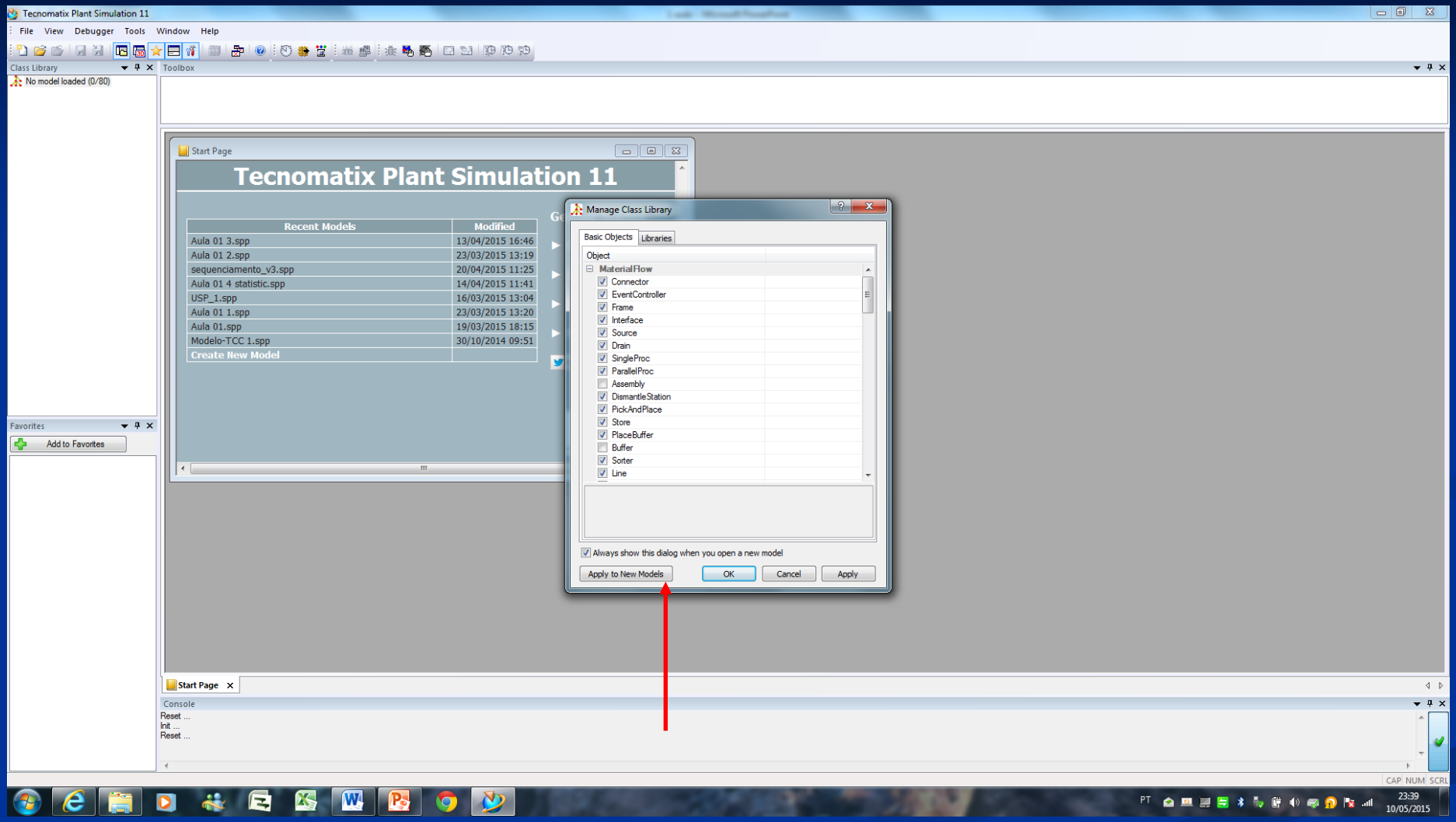
FRAMING

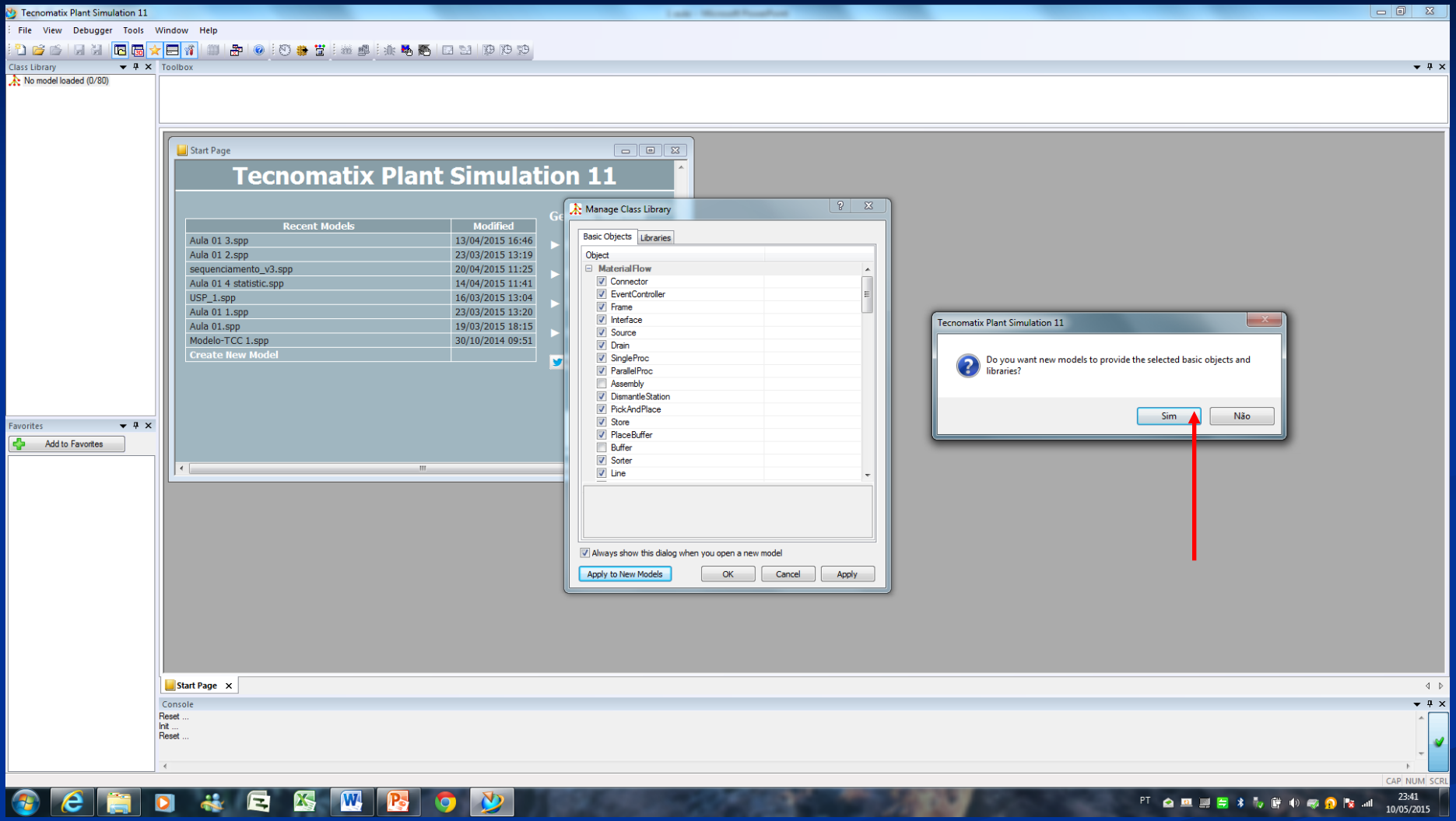

Tecnomatix Plant Simulation 11
File View Debugger Tools Window Help
New
Open...
Close
Save
Save As...
Teamcenter
Send E-mail...
Pack and Go...
Compare With...
Properties...
Show Model History...
Manage Class Library...
1 Aula 01 3
2 Aula 01 2
3 C:\Users\...\sequenciamento_v3
4 Aula 01 4 statistic
5 USP_1
6 Aula 01 1
7 Aula 01
8 C:\Users\...\Modelo-TCC 1
Exit
Add to Favorites
Start Page
Tecnomatix Plant Simulation 11
Recent Models
Modified
Aula 01 3.spp 13/04/2015 16:46
Aula 01 2.spp 23/03/2015 13:19
sequenciamento_v3.spp 20/04/2015 11:25
Aula 01 4 statistic.spp 14/04/2015 11:41
USP_1.spp 16/03/2015 13:04
Aula 01 1.spp 23/03/2015 13:20
Aula 01.spp 19/03/2015 18:15
Modelo-TCC 1.spp 30/10/2014 09:51
Create New Model
Getting Started
Examples/Infos
Demo Videos
Tutorial
New Features
Follow @PlantSimulation
SIEMENS
Start Page
Console
Reset ...
Init ...
Reset ...
Creates a new model
CAP NUM SCRL
PT
23:38
10/05/2015

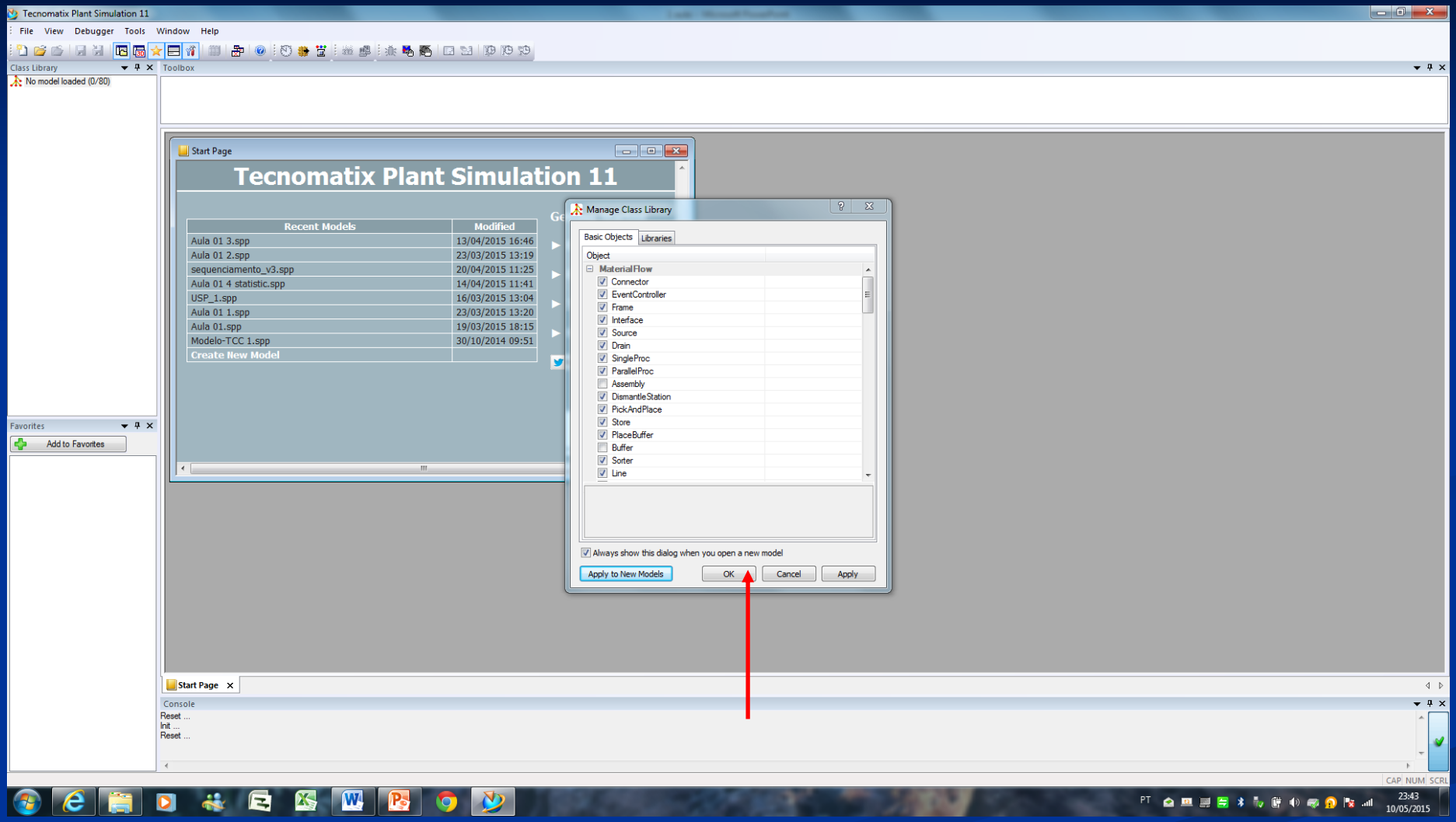
Ou

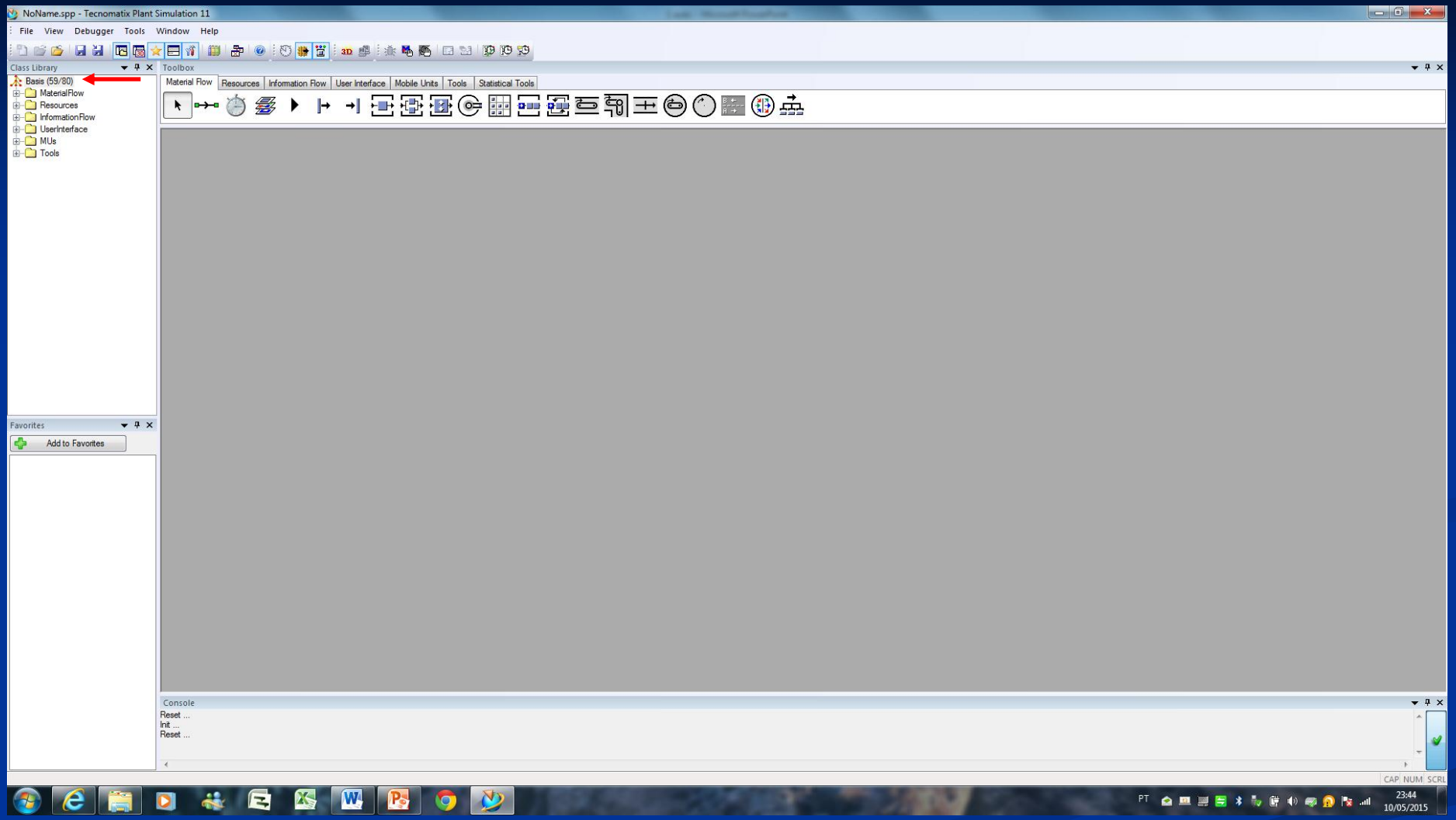
DIRETO NO QUADRO

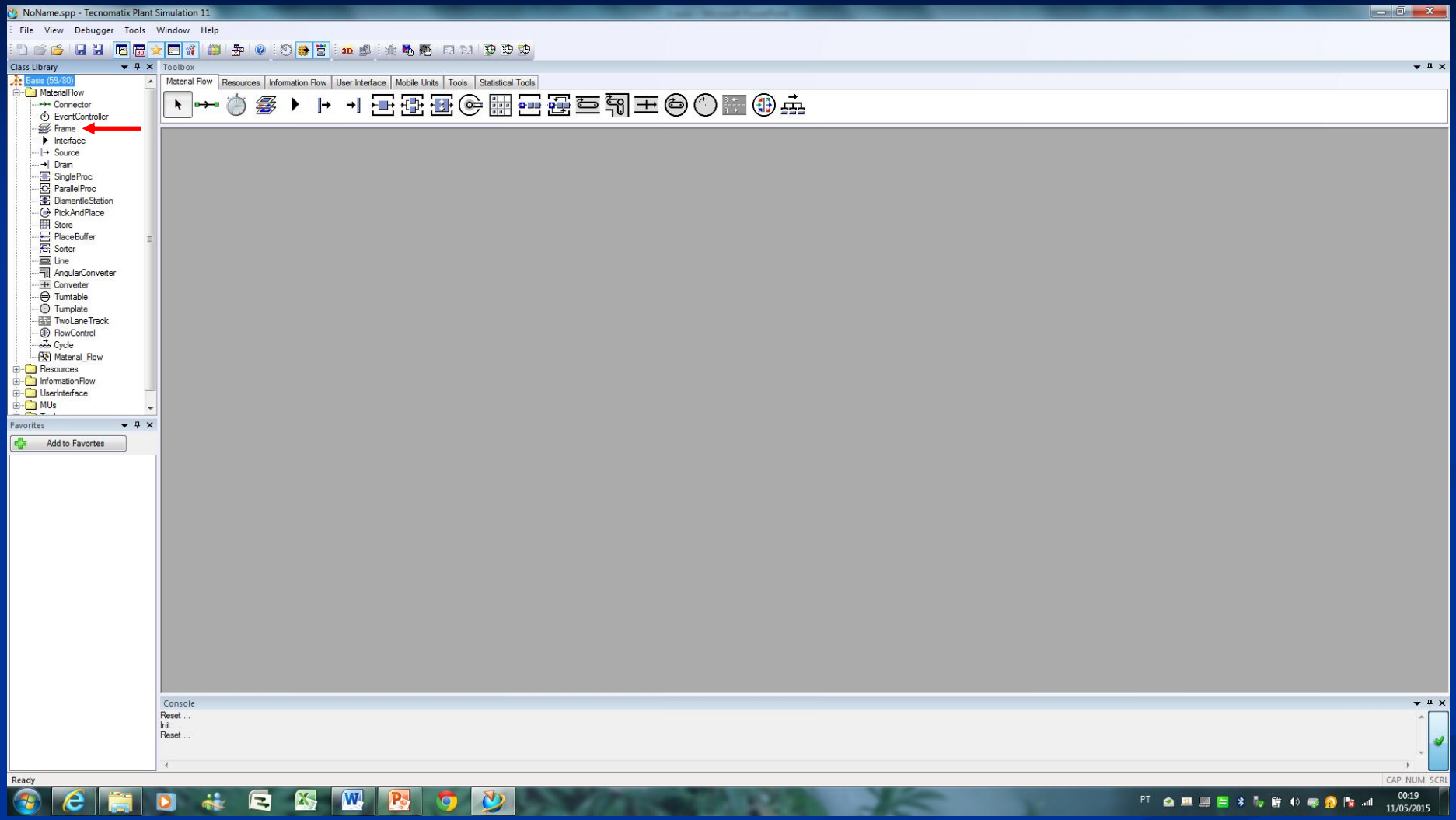

Tecnomatix Plant Simulation 11
File View Debugger Tools Window Help
Class Library
No model loaded (0/80)
Toolbox
Favorites
Add to Favorites
Start Page
Tecnomatix Plant Simulation 11
Recent Models
Modified
Aula 01 3.spp 13/04/2015 16:46
Aula 01 2.spp 23/03/2015 13:19
sequenciamento_v3.spp 20/04/2015 11:25
Aula 01 4 statistic.spp 14/04/2015 11:41
USP_1.spp 16/03/2015 13:04
Aula 01 1.spp 23/03/2015 13:20
Aula 01.spp 19/03/2015 18:15
Modelo-TCC 1.spp 30/10/2014 09:51
Create New Model
Getting Started
Examples/Infos
Demo Videos
Tutorial
New Features
Follow @PlantSimulation
SIEMENS
Start Page
Console
Reset ...
Init ...
Reset ...
newmodel:
CAP NUM SCRL
PT
00:14
11/05/2015

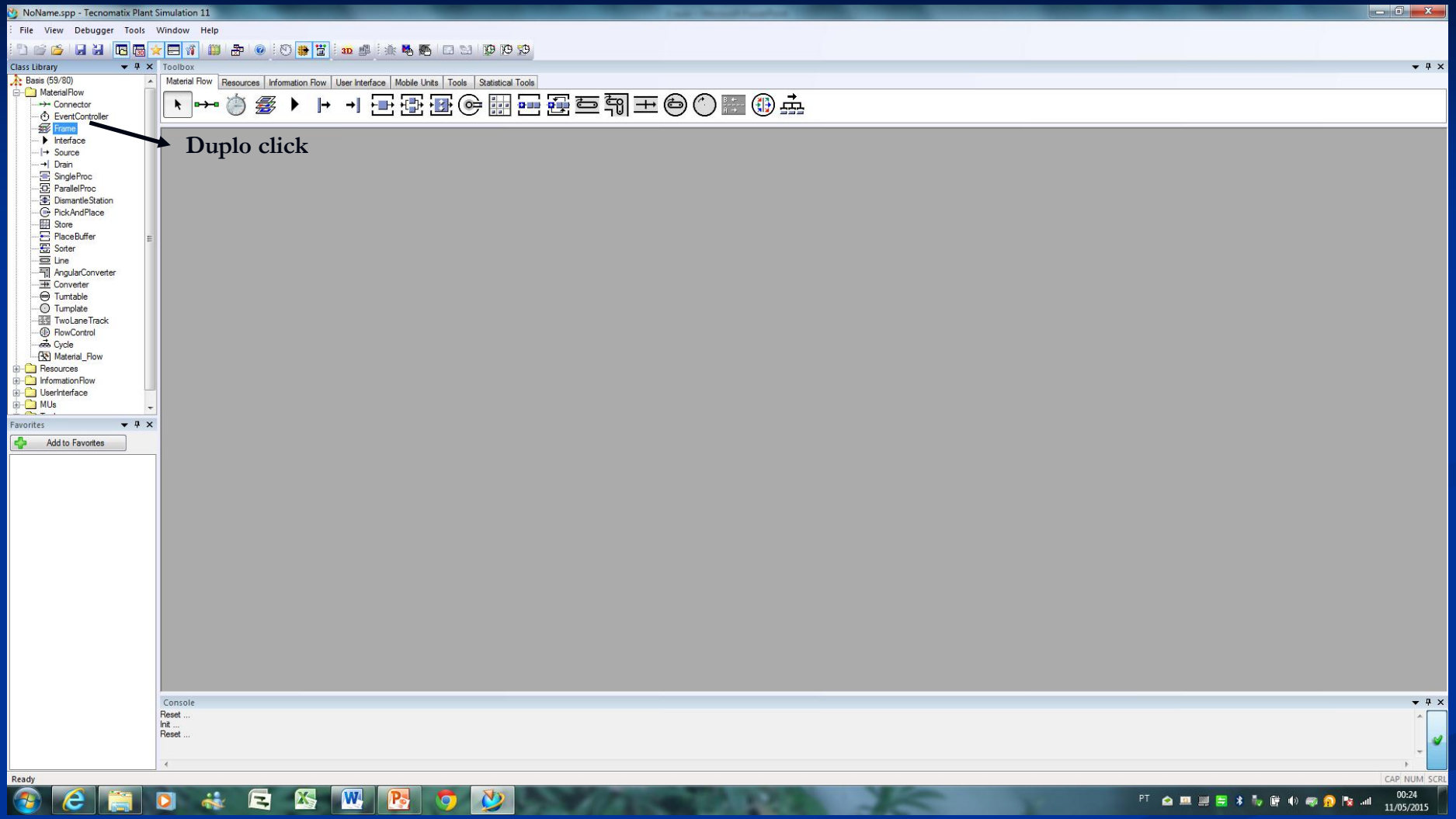

Tecnomatix Plant Simulation 11
File View Debugger Tools Window Help
Class Library
No model loaded (0/80)
Toolbox
Favorites
Add to Favorites
Start Page
Tecnomatix Plant Simulation 11
Recent Models
Modified
Aula 01 3.spp 13/04/2015 16:46
Aula 01 2.spp 23/03/2015 13:19
sequenciamento_v3.spp 20/04/2015 11:25
Aula 01 4 statistic.spp 14/04/2015 11:41
USP_1.spp 16/03/2015 13:04
Aula 01 1.spp 23/03/2015 13:20
Aula 01.spp 19/03/2015 18:15
Modelo-TCC 1.spp 30/10/2014 09:51
Create New Model
Manage Class Library
Basic Objects
Libraries
Object
MaterialFlow
Connector
EventController
Frame
Interface
Source
Drain
SingleProc
ParallelProc
Assembly
DismantleStation
PickAndPlace
Store
PlaceBuffer
Buffer
Sorter
Line
Always show this dialog when you open a new model
Apply to New Models
OK
Cancel
Apply
Start Page
Console
Reset ...
Init ...
Reset ...
CAP NUM SCRL
PT
23:39
10/05/2015

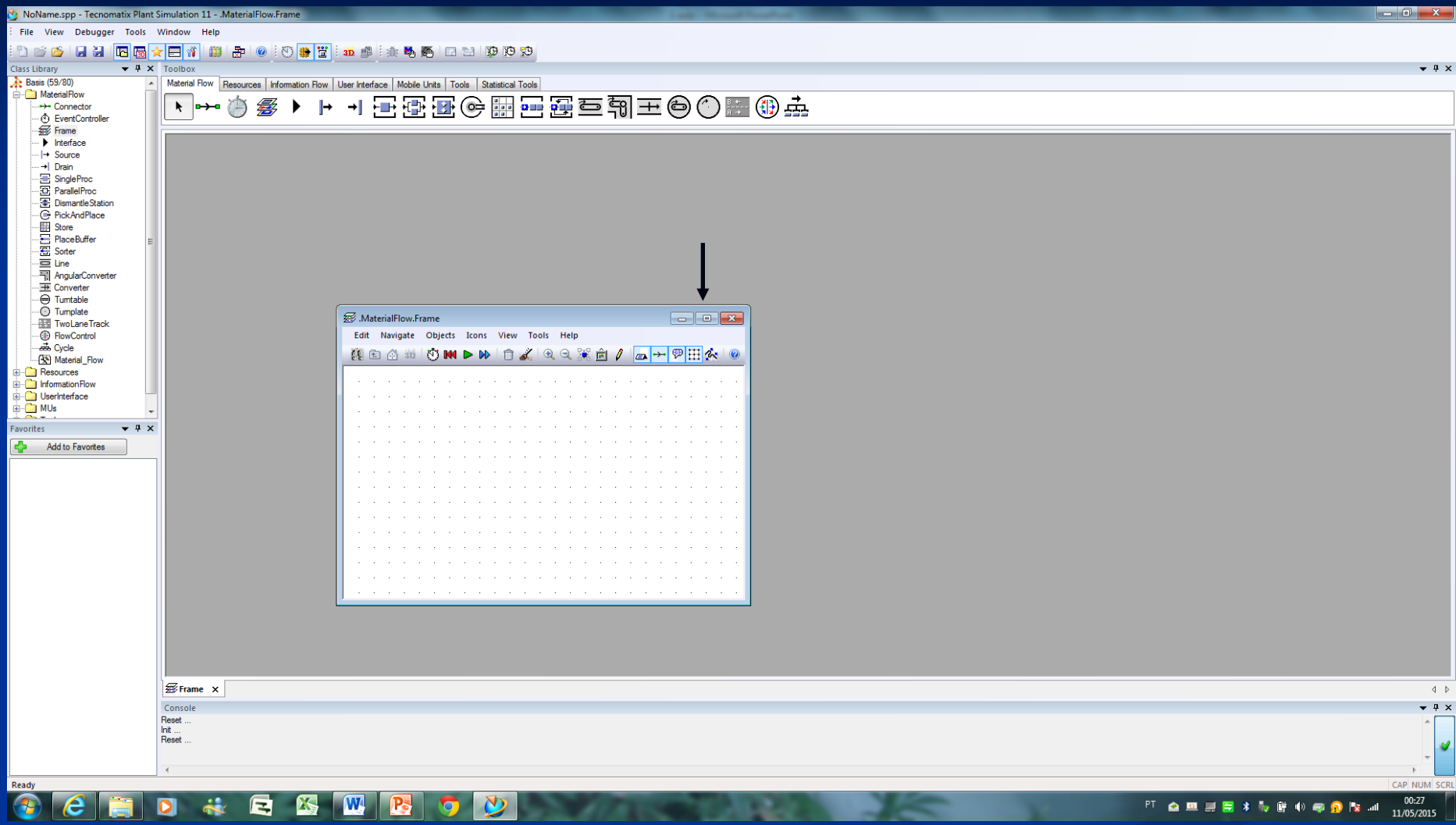

Tecnomatix Plant Simulation 11
File View Debugger Tools Window Help
Class Library
No model loaded (0/80)
Toolbox
Favorites
Add to Favorites
Start Page
Tecnomatix Plant Simulation 11
Recent Models
Modified
Aula 01 3.spp 13/04/2015 16:46
Aula 01 2.spp 23/03/2015 13:19
sequenciamento_v3.spp 20/04/2015 11:25
Aula 01 4 statistic.spp 14/04/2015 11:41
USP_1.spp 16/03/2015 13:04
Aula 01 1.spp 23/03/2015 13:20
Aula 01.spp 19/03/2015 18:15
Modelo-TCC 1.spp 30/10/2014 09:51
Create New Model
Manage Class Library
Basic Objects
Libraries
Object
MaterialFlow
Connector
EventController
Frame
Interface
Source
Drain
SingleProc
ParallelProc
Assembly
DismantleStation
PickAndPlace
Store
PlaceBuffer
Buffer
Sorter
Line
Always show this dialog when you open a new model
Apply to New Models
OK
Cancel
Apply
Tecnomatix Plant Simulation 11
Do you want new models to provide the selected basic objects and libraries?
Sim
Não
Start Page
Console
Reset ...
Init ...
Reset ...
CAP NUM SCRL
PT
23:41
10/05/2015

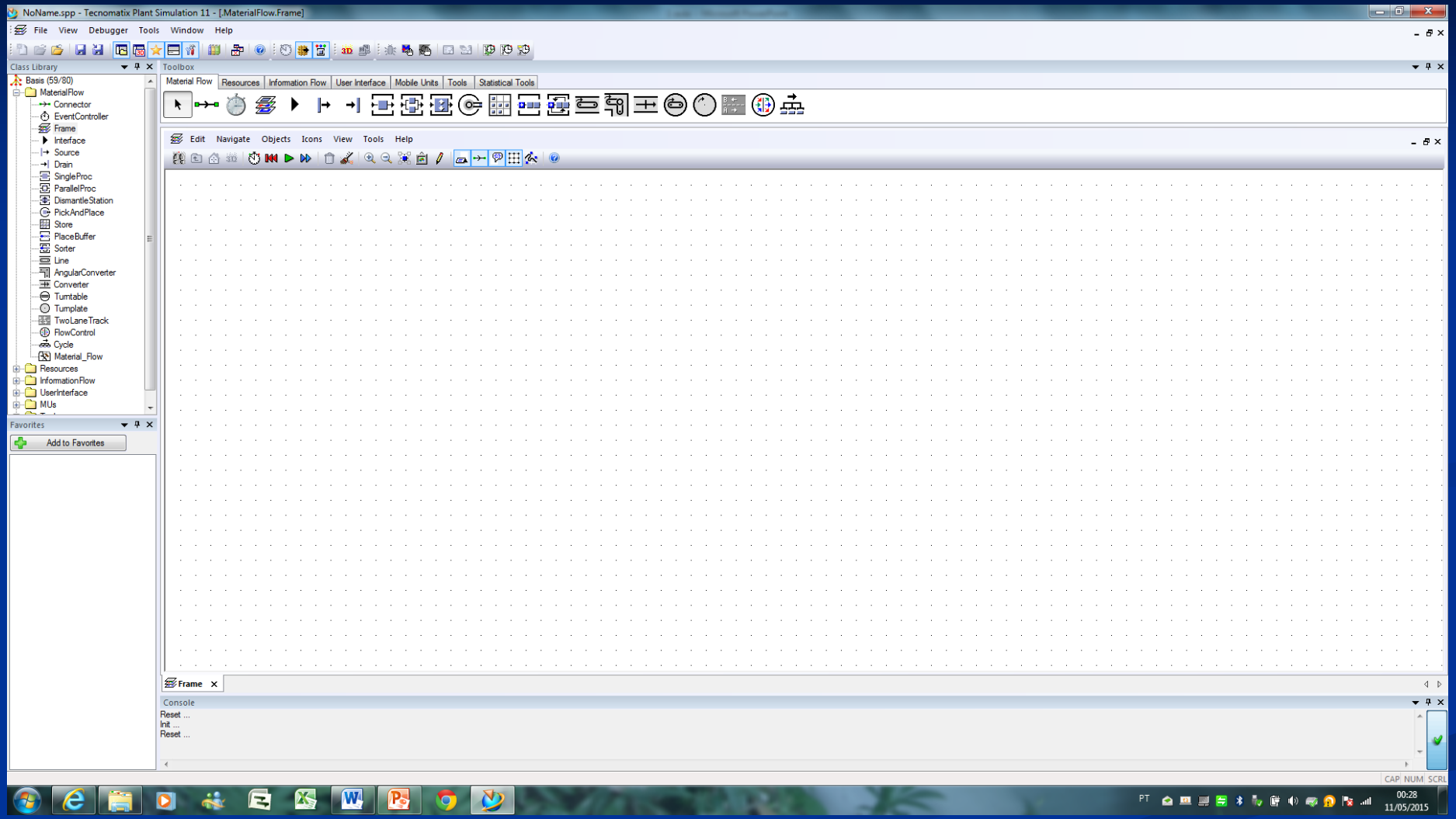

Tecnomatix Plant Simulation 11
File View Debugger Tools Window Help
Class Library
No model loaded (0/80)
Toolbox
Favorites
Add to Favorites
Start Page
Tecnomatix Plant Simulation 11
Recent Models
Modified
Aula 01 3.spp 13/04/2015 16:46
Aula 01 2.spp 23/03/2015 13:19
sequenciamento_v3.spp 20/04/2015 11:25
Aula 01 4 statistic.spp 14/04/2015 11:41
USP_1.spp 16/03/2015 13:04
Aula 01 1.spp 23/03/2015 13:20
Aula 01.spp 19/03/2015 18:15
Modelo-TCC 1.spp 30/10/2014 09:51
Create New Model
Manage Class Library
Basic Objects
Libraries
Object
MaterialFlow
Connector
EventController
Frame
Interface
Source
Drain
SingleProc
ParallelProc
Assembly
DismantleStation
PickAndPlace
Store
PlaceBuffer
Buffer
Sorter
Line
Always show this dialog when you open a new model
Apply to New Models
OK
Cancel
Apply
Start Page
Console
Reset ...
Init ...
Reset ...
CAP NUM SCRL
PT
23:43
10/05/2015


NoName.spp - Tecnomatix Plant Simulation 11
File View Debugger Tools Window Help
Class Library
Basis (59/80)
MaterialFlow
Resources
InformationFlow
UserInterface
MUs
Tools
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Statistical Tools
Console
Reset ...
Init ...
Reset ...
CAP NUM SCRL
PT
23:44
10/05/2015


NoName.spp - Tecnomatix Plant Simulation 11
File View Debugger Tools Window Help
Class Library
Basis (59/80)
MaterialFlow
Connector
EventController
Frame
Interface
Source
Drain
SingleProc
ParallelProc
DismantleStation
PickAndPlace
Store
PlaceBuffer
Sorter
Line
AngularConverter
Converter
Turntable
Tumplate
TwoLaneTrack
FlowControl
Cycle
Material_Flow
Resources
InformationFlow
UserInterface
MUs
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Statistical Tools
Console
Reset ...
Init ...
Reset ...
Ready
CAP NUM SCRL
PT
00:19
11/05/2015


NoName.spp - Tecnomatix Plant Simulation 11
File View Debugger Tools Window Help
Class Library
Basis (59/80)
MaterialFlow
Connector
EventController
Frame
Interface
Source
Drain
SingleProc
ParallelProc
DismantleStation
PickAndPlace
Store
PlaceBuffer
Sorter
Line
AngularConverter
Converter
Turntable
Tumplate
TwoLaneTrack
FlowControl
Cycle
Material_Flow
Resources
InformationFlow
UserInterface
MUs
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Statistical Tools
Duplo click
Console
Reset ...
Init ...
Reset ...
Ready
CAP NUM SCRL
PT
00:24
11/05/2015


NoName.spp - Tecnomatix Plant Simulation 11 - .MaterialFlow.Frame
File View Debugger Tools Window Help
Class Library
Basis (59/80)
MaterialFlow
Connector
EventController
Frame
Interface
Source
Drain
SingleProc
ParallelProc
DismantleStation
PickAndPlace
Store
PlaceBuffer
Sorter
Line
AngularConverter
Converter
Turntable
Tumplate
TwoLaneTrack
FlowControl
Cycle
Material_Flow
Resources
InformationFlow
UserInterface
MUs
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Statistical Tools
.MaterialFlow.Frame
Edit Navigate Objects Icons View Tools Help
Frame
Console
Reset ...
Init ...
Reset ...
Ready
CAP NUM SCRL
PT
00:27
11/05/2015


NoName.spp - Tecnomatix Plant Simulation 11 - [.MaterialFlow.Frame]
File View Debugger Tools Window Help
Class Library
Basis (59/80)
MaterialFlow
Connector
EventController
Frame
Interface
Source
Drain
SingleProc
ParallelProc
DismantleStation
PickAndPlace
Store
PlaceBuffer
Sorter
Line
AngularConverter
Converter
Turntable
Turnplate
TwoLaneTrack
FlowControl
Cycle
Material_Flow
Resources
InformationFlow
UserInterface
MUs
Favorites
Add to Favorites
Toolbox
Material Flow
Resources
Information Flow
User Interface
Mobile Units
Tools
Statistical Tools
Edit Navigate Objects Icons View Tools Help
Frame
Console
Reset ...
Init ...
Reset ...
CAP NUM SCRL
PT
00:28
11/05/2015

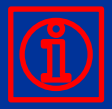
ANEXO II

ANONYMOUS IDENTIFIER

anonymous identifier

anonymous identifier @

- O identificador anônimo @ (at) aponta a MU que desencadeou o controle. Quando você controla a entrada ou a saída de um objeto no fluxo de materiais, o identificador anônimo @ permite que você acesse a MU que entrou ou está pronta para sair do objeto.
- A atribuição é única, mesmo se vários métodos de controle são acionados ao mesmo tempo durante a simulação, o *Plant Simulation* sempre processa por completo a MU antes de prosseguir para a próxima MU.
- Example: @.move(parallelproc.succ(3));



Anonymous Identifiers identificadores anônimos

Normalmente, o nome de um controle que você programou em um método não altera em seu modelo. O caminha do método altera de modelo para modelo, contudo. Se um controle requer o caminho atual, você pode checar usando um identificador anônimo (***anonymous identifiers***). Os identificadores anônimos (***Anonymous identifiers***), tal como ***@***, ***basis***, ***current***, ***?***, ***root***, ***rootFolder*** ***e*** ***self*** fazer uso de um método mais flexível e independente do seu contexto. Você pode inserir em diferentes modelos sem ter que modificar seu código fonte (***source code***).

ANEXO III

ARRAY

Array

SimTalk fornece tipos de dados array (***array data types***). Um ***array*** é campo de valor uni-dimensional ou bi-dimensional de um dos tipos de dados da base listados acima sob os (***Data Types***).

Base data types pode ser todos tipos de dados exceto par a Tabela, lista, ***stack*** e ***queue***. Quando você entrar com algum como o tipo base de dado (***the base data type***), cada item do campo do valor pode ter um tipo diferente de dado (***different data type***). Então, mesmo listas e tabelas podem ser colocados ***na matriz (array).***

Array indexa uma única base (***indexes are one-based***), por exemplo, elas começam em 1, não em 0.

Você pode declarar variáveis ***array*** do seguinte modo:

1) a : integer[10];
2) b : boolean[10,20];
3) a : string[];

Array

When you enter a single number within the square brackets, then it is a one-dimensional array of fixed size, for example a : integer[10];.

When you enter a two numbers within the square brackets, then it is a two-dimensional array of fixed size, for example b : boolean[10,20];.

When you do not enter any number at all within the square brackets, then it is a one-dimensional array, whose size is not fixed, and which is empty initially, for example a : string[];.

Compare this example:

```
is
   vector3  : real[3];    -- one-dimensional array with 3 real values
   matrix3x3 : real[3,3]; -- two-dimensional array with 9 real values
   objList : object[]; -- one-dimensional array with n objects (the size can change)
   a        : any[];      -- one dimensional array with n values of any data type
do
   print vector3[2]; -- prints 0 on the console
   vector3 := makeArray(1.0, 2.0, 3.0); -- fill a one-dimensional array
   print vector3[2]; -- prints 2 on the console
```

Array

```
-- For filling a two-dimensional array you might enter:
for local x := 1 to matrix3x3.xDim loop
   for local y := 1 to matrix3x3.yDim loop
      matrix3x3[x,y] := x + y;
   next;
next;
print matrix3x3; -- prints [2, 3, 4][3, 4, 5][4, 5, 6]

print objList.dim;  -- prints 0 on the console
objList.append(SingleProc); -- add SingleProc to the array
print objList.dim;  -- prints 1 on the console
print objList[1];   -- prints the path to the SingleProc on the console
a := vector3;
a.append("Hello World");
print a.dim;        -- prints 4 on the console
print a;            -- prints [1, 2, 3, Hello World] on the console
end;
```

ANEXO IV

OBJECT

OBJECT

Usamos o termo objeto para representar u determinado elemento do mundo real. Mas comente analisaremos os objetos que tem relevância para a solução de um determinado problema. Portanto, o objeto é uma entidade do mundo real que merece representação para o ambiente estudado. Objetos são instâncias de classes, que determinam qual informação um objeto contém e como ele pode manipulá-la. É uma entidade capaz de reter um estado (informação) e que oferece uma série de operações (comportamento) ou para examinar ou para afetar este estado. É através deles que praticamente todo o processamento ocorre em sistemas implementados com linguagens de programação orientada a objetos.

OBJECT

Uma variável do tipo de dado ⇒ "objeto" (***A variable of data type object***) aponta para um objeto do seu modelo de simulação ou não. Você pode salvar algum objeto representado graficamente como o "***data type object***". Esta variável pode sre uma variável local, um parâmetro formal, uma variável global e valores em tabela.

Você pode atribuir tanto objetos quanto *string*s para as variáveis do "***data type object***" mencionado acima. Quando você atribuir um objeto, o *Plant Simulation* cria um *Object Reference*, independentemente se você especificar o objeto com um caminho absoluto ou similar. Quando você especificar o objeto com um caminha alternativo, o *Plant Simulation* imediatamente resolve o caminha similar dentro do contexto do método atual e caria um *object reference* para o objeto calculado.

Vamos supor que o método ***.Models.Frame.method*** contem a seguinte chamada do método: ***SubFrame.methodU(SingleProc)***.

OBJECT

- O ***Plant Simulation*** resolve o caminho similar ***SingleProc*** para um ***object reference*** para ***.Models.Frame.SingleProc***, e passa este ***object reference*** para o método dentro do ***SubFrame***. Se o caminho não pode ser resolvido para um objeto, o ***Plant Simulation*** abre o ***Debugger***.
- When a string is passed to a variable, the string is assigned as a path to the variable. Regardless if this path is relative or absolute, Plant Simulation does not resolve it at this point in time. Instead, the path is resolved each and every time when the variable is read.
- Let’s suppose that the method .Models.Frame.method contains the following method call: SubFrame.methodU("SingleProc");
- Plant Simulation passes the relative path SingleProc to the method within the SubFrame. At this point in time such an object neither has to exist within the context of the calling method, nor within the context of the called method as the path is not resolved yet. Let’s suppose that we inserted an object of type SingleProc into the SubFrame and that this object has the name SingleProc1. Let’s also suppose that the source code of methodU in the SubFrame looks like this:

- The Absolute Path
- The absolute path is formed by adding up all names beginning with the top-level folder and the Frame in the Class Library until it reaches the object of your choice.
- The absolute path starts with a period—standing for the Class Library—followed by the name of the folder and top-level Frame. Then a period and a name alternate until you reach the desired object, which again is preceded by a period:
- .Models.MyPlant.Engine_Assembly_Anytown.MyStation
- You might, for example, use the absolute path for a control method. This method might always execute certain actions before starting a simulation run, and which never change. Then it makes sense to add this method to the Class Library and reference it using its absolute path that never changes.
- If, on the other hand, you want to use the same method, but with different source code, in instantiated Frames, you will use The Relative Path, to make sure that each Frame finds the method where you programmed actions specific to this Frame.

- The Relative Path
- The relative path starts in the current name space—formed by all objects located within one Frame —or the Frame in which the Method object is located.
- The relative path starts with an anonymous identifier, a name in the current name space or a built-in method. Then a period follows and the name of a Frame or a period and a built-in method. The last entry is a period and the name of the object or a period and a built-in method or a period and an attribute.
- This table lists, where the relative path starts for the different objects:

	User-defined attribute created for an MU	User-defined attribute created for an object	User-defined attribute created for a Frame	Built-in objects
Object variable	In the Frame in which the material flow object, on which the MU stays, is located.	In the Frame in which the material flow object is located.	Within that Frame.	In the Frame in which the object is located.
Method (current)	In the Frame in which the material flow object, on which the MU stays, is located.	In the Frame in which the material flow object is located.	Within that Frame.	In the Frame in which the object is located.
Table with column of data type object	The material flow object, on which the MU stays.	In the Frame in which the material flow object is located.	In the Frame into which you inserted that Frame.	In the Frame in which the object is located.

- For user-defined attributes of static objects, the Frame in which the object is located is the starting point of the relative path.
- For user-defined attributes of the MU the static object on which the MU is located is the starting point of the relative path, independent of the fact that the location of the MU might be another MU.
- Suppose that in the Method object M, which you inserted into a Frame, you would like to access the object named MA by employing a relative path. The Method object is located in Frame W. The object MA is not located in the same name space as M and has to be identified by an additional path. The object MA is located in the Frame that also contains the method M. The relative path begins with the name of the Frame in which the object is located, followed by a period and the object name.
- Example: variable := AP.MA;
- In our second example the Method M is supposed to access the Store located in Frame L and to query the number of MUs. The Store is not located in the same Frame as the Method. Therefore, we will use an anonymous identifier.
- The keyword current returns the current Frame. The method location/ ~ returns the Frame located above in the hierarchy of objects. From now on you can access L and the Store.
- Example: Var := current.location.L.store.numMU
- Var := location.L.store.numMU

ANEXO V

CLASS LIBRARY

Class Library

- The Class Library shows the built-in objects in a hierarchical view grouped in folders and subfolders. By default the Class Library contains a folder for ***Material Flow Objects***, for ***Resource Objects***, for ***Information Flow Objects***, for ***Mobile Objects***, for ***Display and User Interface Objects***, for ***Models***, and for ***Tools***.
- It also shows the objects of application object libraries which you purchased. When you modeled your own objects, either in ***Frames*** or by modifying any of the built-in objects, the Class Library also shows these objects. We call these objects application objects. You find examples of application objects in the folder Tools: The ***ExperimentManage***r, the ***TransferStation***, the ***SankeyDiagram***, etc.
- When your simulation model also has a 3D part, the 3D Library shows the viewer objects that correspond to the objects in your 2D part of the model.

ANEXO VI

SIMTALK

STATEMENTS - DECLARAÇÕES

SimTalk

- A linguagem de programação *SimTalk* amplia as possibilidades que você pode modelar e controlar a sua simulação. Cada objeto tem propriedades próprias embutidas que provêm aspectos muito úteis. Quando seu modelo requer mais detalhes ou propriedades complementamente diferentes, você deve programar estes na linguagem de programação *SimTalk*.
- Primeiro, você deve entrar com as declarações (***statements***), as quais o programa de compilação ou de interpretação faz uso (***the built-in Interpreter program executes***) no objeto método (***object Method***). Você pode também combinar o objeto método com os objetos do fluxo de materiais e objetos do fluxo de informação (***Method object with the material flow objects and the information flow objects***) para criar modelos de grande complexidade. Então pressione as teclas F7 e F5 para executar seu código fonte (***source code***).
- Quando você executar a simulação, o interpretador executa o código fonte (***source code***), escrito no objeto método linha a linha e (***Method line-by-line***) assume as ações que o código fonte representa.

Object Method (objeto ⇒ método)

Programando um Método (***Programming a Method***)

- O objeto método (***The object Method***) é o contenedor de pequenos programas, os quais você escreveu. Para abrir esses programas e acessar o código fonte (***source code***) a ser executado, dê um ***double-click*** em seu ícone no ***Frame*** no qual você o inseriu. Mover o ponteiro após a palavra chave ***do***, pressione o enter e acione o código fonte (***source code***).

Object Method (objeto ⇒ método)

Programando um Método (***Programming a Method***)

- A estrutura de um método é dividida em algumas partes. Se você não necessita de uma determinada parte, você pode deletar ou omitir. Cada método é estruturado do seguinte modo:

[parameters]

[data type of the return value]

is

[local variables]

do

[source code]

end;

Um método pode, por exemplo, ser parecido com este:

```
-- My Tooltip. My text, my text, my text, ...
(aSingleProc : object)              -- parameter
: boolean                           -- data type of the return value
is
do                                  -- start of a block of source code statements
      if aSingleProc.occupied then  -- start of the if-statement
            aSingleProc.cont.move;
            return true;
       else
            return false;
       end;                         -- end of the if-statement
end;                                -- end of the Method
```

Result of Function Value of a Method

Mathematical functions return a result. A Method may also return a result. To make a method work like a function, add a colon (:) and a data type before the keyword is. Also enter additional parameters in front of the colon (:).

* A function only returns one result.

Within the function you have to assign the result to the return value result. After the function has been executed, Plant Simulation returns the contents of this variable to the caller.

Compare these examples:

- A function without a parameter, calculating a real value, looks like this:

: real

is

Result of Function Value of a Method

- A function with a parameter, returning a string value, looks like this:

(CustomerNo:integer)

: string

is

- A function with several parameters, returning a table, looks like this:

(orders,deliveries:list; delicery_date:time)

:table

is

Result of Function Value of a Method

Instead of using result, you can also terminate the method using return, compare Exiting Methods Using Return. The data types stack, queue, list, and table things behave different. At the beginning, the local variable result is empty. Before accessing result you either assign a value to result or use the method create.

```
-- returns a list that contains the value added tax amount
-- added to the prices of the price list that was passed
(pricelist : queue; VAT : real) : list[real]
is
   i : integer; -- loop index
do
   result.create; -- instantiates the results list
   i := 1;
   while not pricelist.empty loop -- stop condition
      result[i] := pricelist.pop * (1 + VAT);
      i := i + 1; -- increases the index
   end;
end; -- end of function
```

Result of Function Value of a Method

create

Syntax: <local_variable>.create[(NumberOfRows:<integer>)];

The method create creates an empty data structure without contents in the local variable designated by <local_variable>.

* The method create applies to local variables of data type List, Stack, Queue and Table.

Parameter

The optional parameter NumberOfRows of data type integer designates the number of rows in the list or table.

Example: is

```
orderlist : table[string,real];
do
  orderlist.create;
  orderlist[1,1] := "cans";
  orderlist[2,1] := 3000.0;
  orderlist.forget;    -- destroys the table
  orderlist.create(4); -- recreates the table with 4 rows
end;
```

Related Topic

Data Types in Local Variables

Result of Function Value of a Method

Data Types in Local Variables

You can use these data types in a local variable: Integer, Real, Length, Speed, Acceleration, Weight, Time, Date, DateTime, Boolean, String, Object, Table, List, Stack, Queue, and Any.

For the data types stack, queue, list and table you have to enter the data type or types of the columns in between brackets, [data type]. These local variables slightly differ from the objects, which you insert into a Frame. As opposed to the objects, they only provide built-in methods for Instantiation, State, Access and Order.

Data type	Shares built-in properties of the object
List	CardFile
Queue	QueueFile
Stack	StackFile
Table	TableFile

* Before first accessing these data types, you have to apply the method create to the variable or you have to assign a value to the variable.

Result of Function Value of a Method

Example: is

```
  l : list[string];
do
  l.create;
  l.insert(1,"Hello");
end;
```

Plant Simulation automatically initializes all local variables. The initial value depends on the data type:

integer, real, length, ...	0
boolean	false
string	""
object	void
time	0:00:00

Result of Function Value of a Method

Local Variables

If you want to reuse the result of a calculation again at a later point in time, you can save the result in a local variable. Local variables can only be accessed within the Method in which you declared it, i.e., they are only available locally within this Method. You can use a local variable if you do not need the value again after the method call.

If you need to access the saved value later on, use the object Variable instead.

The name of a local variable is only known in the Method in which you declare it. Other Methods cannot access this local variable. For each call Plant Simulation creates a new set of local variables. For example, when a Method calls itself, Plant Simulation creates a new set of local variables and the Method may only access this new set. Only when the call is finished, the previous local variables are available again with the previous values.

You have to declare a local variable before you can use it. Basically, you can accomplish this in one of two ways: You can:

- Declare a Local Variable at the Beginning of the Source Code
- Declare a Local Variable Anywhere within the Source Code

Before you assign a value to the local variable, it has an initial value. It is 0 for all numerical data types, false for the data type boolean, empty string ("") for the data type string, and void for the data types Object and List/Table. The initial value for the data types Date and DateTime depends on the Plant Simulation version in which you initially created your simulation model.

* We selected the value for the initial value for the data types date and dateTime to be close to the start date entered on the in The figure the EventController to reduce rounding errors.

Result of Function Value of a Method

Declare a Local Variable Anywhere within the Source Code

You can also declare a local variable anywhere within the source code. Start to declare your variable with the keyword local, followed by one or several identifiers for the local variables you are defining. After that you can enter a colon and define the data type of the local variable.

Example: local x, y : integer;

A local variable, which you declare like is visible from the line, in which you defined it, to the end of the source code. Just like a local variable, which you declared between is and do, a local variable defined via local is initialized with an initial value. Compare Data Types in Local Variables.

You can also assign a value to the local variable when you declare it.

Example: is

```
do
   local a : length := Track1.length;
   local b : length := Track2.length;
   local c : length := sqrt(a*a + b*b);
   print c;
end;
```

Result of Function Value of a Method

When you assign a value to the local variable while you declare it, declaring the data type is optional. The data type of the variable is then determined by the assigned value.

Example: is

```
do
   local a := Track1.length;
   local b := Track2.length;
   local c := sqrt(a*a + b*b);
   print c;
end;
```

In the example above the variables a and b will take the data type length. The variable c will take the data type real, as the function sqrt returns a value of data type real.

For-Loop with a Local Loop Variable

As a special case you can use the keyword local in a for-loop to define the loop variable. The loop variable is then only visible within the loop. Outside of the loop the identifier instead points to an object with this name if this object exists, for example to a global variable.

Result of Function Value of a Method

```
Example:  is
do
   -- the loop variable 'i' is only visible inside the loop
   for local i := 1 to ParallelProc.numMU loop
      print ParallelProc.mu(i);
   next;
   print i; -- the loop variable is not visible here, so 'i' must be the name of an object
   -- another loop with its own loop variable 'i'
   for local i := ParallelProc.numMU downto 1 loop
      print ParallelProc.mu(i);
   next;
   -- the local variable 'i' is visible from here to the end of the source code
   local i := 1.234;
   print i; -- prints 1.234

   for local i := 1 to 10 loop -- does not compile, because 'i' is already declared
   next;
end;
```

Result of Function Value of a Method

Declare a Local Variable at the Beginning of the Source Code

Enter the name of your local variable to be declared after the keyword is, followed by a colon and the data type which the variable is to have. A semicolon terminates the declaration, followed by the word do.

If you would like to declare several variables of the same type, separate their individual names with commas, followed by a colon, the data type and a semicolon.

Example: is

```
   index : integer;    -- single declaration
   VAT, loctax : real; -- two variables with the same data type
   customers : list;   -- customer list
   orders : table[integer,string,real]; -- table with three columns
   complaints : queue[time];           -- queue of data type time
do
   -- your source code
end;
```

Result of Function Value of a Method

Constants and Variables

Tecnomatix Plant Simulation provides Constants, Local Variables, Global Variables, and Parameters allowing you to set values. A constant has a fixed value, while the value of a variable can change over time.

Parameters

You can pass value to a Method, when this Method is called. These values are called arguments. You then have to declare the same number of parameters within the Method and these parameters have to have the same data types as the passed arguments. An exception are integer and real values. Plant Simulation automatically converts these.

- When converting real values to integer values, Plant Simulation deletes numbers after the decimal point. This may cause your method to behave not as you would expect. In general it is a good idea to avoid automatic type conversion.

Global Variables

By definition any object in the model can access a global variable, which is a non-fixed value. Data stored in a global variable by a method will still be present when Plant Simulation terminates the execution of the method. Use global variables (or entries in lists) to keep data for extended periods of time during a simulation run.

Plant Simulation provides the object Variable as a global variable. any method may access it by its name and an absolute or relative path pointing to it.

Result of Function Value of a Method

Constants

Constants have a fixed value. Plant Simulation provides these constants:

boolean	string
integer	pi
real	

ANEXO VII

CREATE

Create

create

Syntax: <MU_path>.create(Location:<mu_location> [,Position:length, CopyStatistics:boolean]);

O método ***create*** cria uma instância da classe MU designada por ***<MU_path>*** sobre um objeto do fluxo de material designado pela parâmetro ***mu_location***. Note que você não pode criar uma MU sobre as estações individuais de um ***Sorter***.

Parâmetros

- A escolha do parâmetro opcional (a partir do seletor de escolha) de localização do ***data type*** comprimento designa a posição do objeto móvel (***mobile object***) sobre um comprimento orientado ao objeto (***on a length-oriented object***).
- O parâmetro opcional da posição do tipo de conjunto de dados *boolean* (***data type boolean***) se as estatísticas das peças é para ser copiada (***true***) ou não (***false***). Se as estatísticas são copiadas, a MU criada tem o mesmo tempo de vida quanto o objeto original após ter sido criado. Se as estatísticas não são copiadas, uma nova MU deve ser criada a qual começa seu tempo de vida a partir deste ponto no tempo.

Se você não especificar o parâmetro, as estatísticas não são copiadas (***false***).

Copying statistics somente é possível, quando o método ***create*** é chamado para uma instância MU.

Create

⇒ ***Example: .part.create(ParallelProc)***;

.part.create(Store[2,5]);

.part.create(Track,9.3);

Track – localizar

SingleProc1.cont.create(SingleProc2, 0, true); -- **duplicar** a MU que está localizada no ***SingleProc1***

O método aplica-se às classes MU e instâncias MU. Neste último caso, o *Plant Simulation* cria uma cópia da MU, incluindo todos os seus atributos. Note-se que nenhuma herança existe entre o MU original e a cópia.

Uma MU criada sobre nos objetos (***point-oriented objects SingleProc e ParallelProc***) ***SingleProc*** e ***ParallelProc*** pode sair do objeto imediatamente, sem ter sido processado. Uma MU deve ser criada na primeira estação disponível sobre o ***PlaceBuffer***, desde que você não tenha designado uma estação específica.

Sobre os objetos orientados por comprimento (***length-oriented objects***), o *Plant Simulation* cria MUs com sua frentre tão perto da saída do objeto quanto possível. Se o comprimento disponível não é suficiente para receber a MU na sua totalidade, o *Plant Simulation* tenta colocar o compromimento remanescente para o objeto precedente. Se isto não é bem sucedido, o ***create fails***.

* Qualquer método ***ConstructorCtrl*** que você inserir deve ser executado.

Create

Return Value

O valor retornado tem o ***data type object*** e designa a MU, a qual foi criada.

⇒ ***Example:*** -- Um método para ser chamado periodicamente por um geraador deve criar MUs do tipo

-- ***Entity*** no ***SingleProc*** chamado ***Start***. A variável chamada ***unsuccAttempts***

-- ***(counts)*** contagem do número de vezes que as MUs poderiam não ser criadas.

```
is
   item : object;
do
   item := .MUs.Entity.create(Start);
   if item = VOID then -- unsuccessful?
      unsuccAttempts := unsuccAttempts + 1;
   end;
end;
```

ANEXO VIII

CONTENTSLIST

ContentsList

contentsList

Syntax: <Path>.A.contentsList[(ContentsList:<table>)];

<Path>.B.contentsList[(ContentsList:<table>)];

The method contentsList returns the entire contents, i.e., all Transporters which are located on the specified lane of the TwoLaneTrack designated by <Path>.

- *If you specify the optional parameter of data type table, the method writes the objects contained in the Contents list into this list.*

- *If you do not specify the parameter, the method returns an array containing the objects contained in the Contents list.*

Parameter

The optional parameter ContentsList of data type table designates the name of the table. Plant Simulation automatically generates the format of this table and deletes any existing format and contents of the table. The contents list is a table with three columns. The data type of column 1 is object, of the second and third length. Column 1 contains the path to the Transporters, column 2 the start position of the MU part, and column 3 the end position in the length unit you specified. The number of rows matches the number of Transporter parts located on the object.

ContentsList

Return Value

O valor de retorno assume algum tipo de dado (***data type***).

Se você não especificar o parâmetro opcional, o método retorna um ***array*** contendo todas as MUs as quais encontram-se localizadas no ***TwoLaneTrack*** no caminho especificado.

⇒ ***Example: is***

```
do
  MyTwoLaneTrack.A.contentsList(TableFile);
end;
```

⇒ ***Example:*** print TwoLaneTrack.A.contentsList;

-- ***might, for example, return*** [.MUs.Transporter:314, 34.488399, 35.988399][.MUs.Transporter:316, 32.988399, 34.488399][.MUs.Transporter:318, 31.488399, 32.988399][.MUs.Transporter:320, 29.988399, 31.488399] in the Console

Verificar

- Create
- Identificadores anônimos
- semente