



Mercury 4.4.3

Features and Bug Fixes

www.modcs.org



Ajustes da Ferramenta

- ✓ Ajuste de funcionalidades da **simulação SPN**
 - ✓ Métricas do tipo: $E\{\#P0\}$
 - ✓ Marking dependent delays. Ex: IF ($\#SU=2$ AND $\#B>1$): $ST*0.5$ ELSE ST ;
- ✓ Geração de modelos CTMC para o **Mathematica**
- ✓ Cálculo do **MTTA** (mean time to absorption) na CTMC



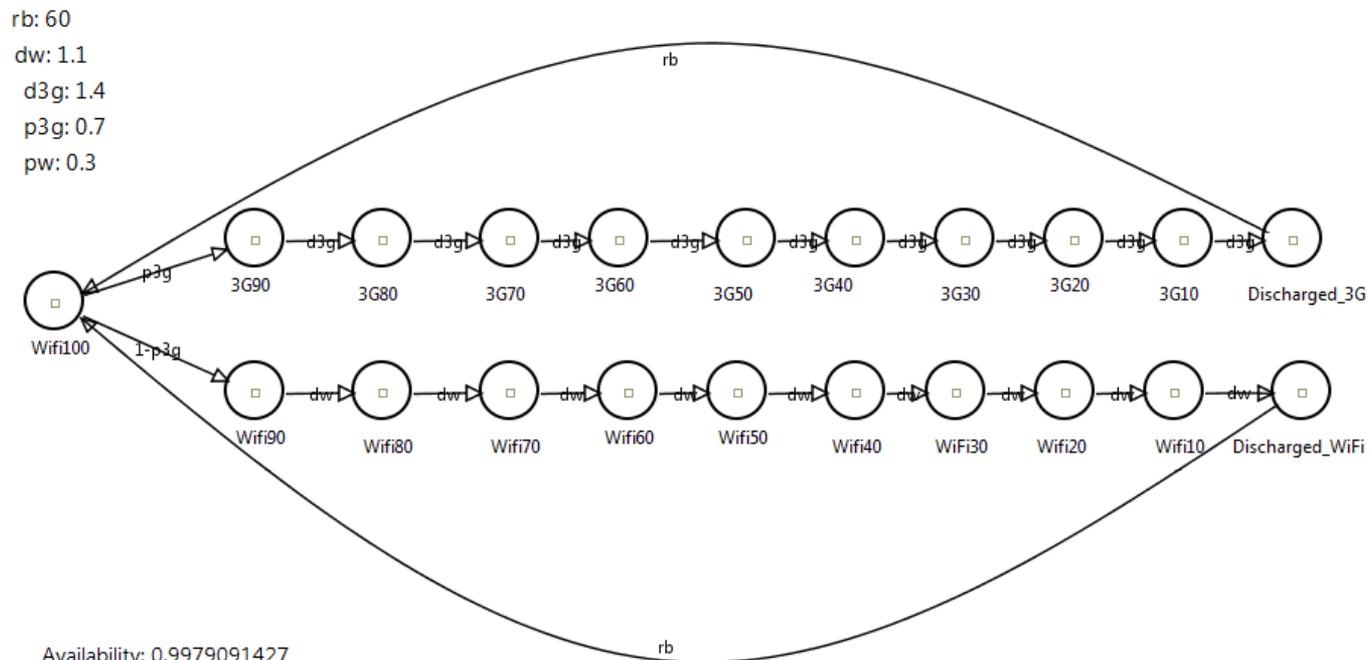
Ajustes da Ferramenta

- ✓ Cálculo da probabilidade de se alcançar um **estado absorvente** em um instante t para CTMCs
- ✓ Inclusão de **métricas** para CTMCs
- ✓ Inclusão de **rewards** para CTMCs
- ✓ Cálculo de **propriedades estruturais** para SPNs

Métricas e rewards na CTMC

Exemplo extraído de:

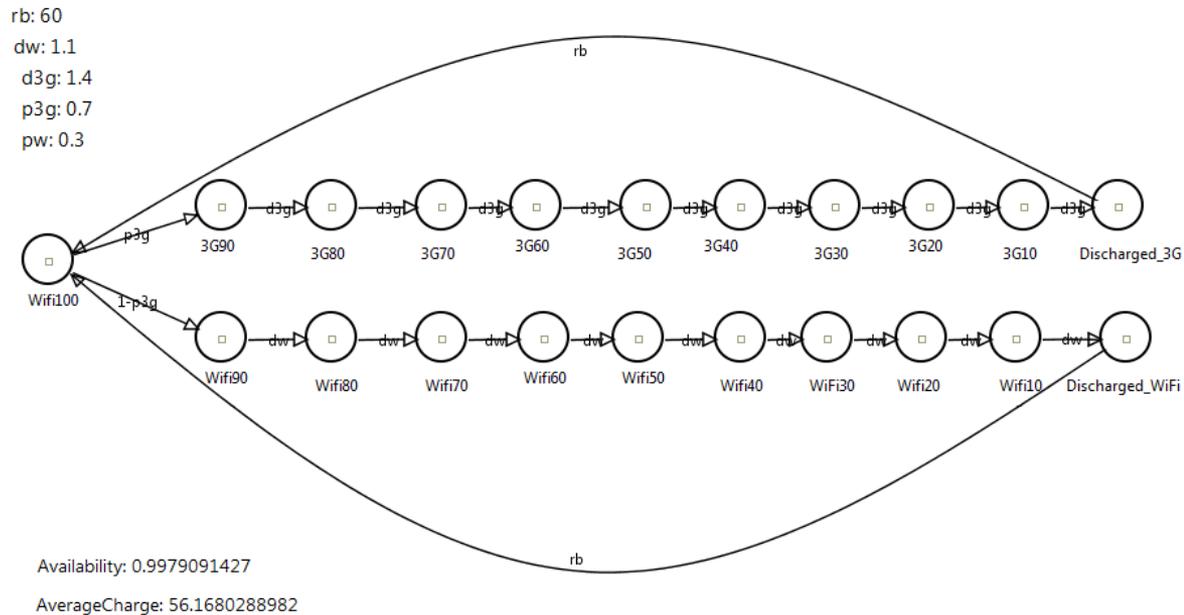
Matos, R.; Araujo, J.; Oliveira, D.; Maciel, P. ; Trivedi, K. **Sensitivity analysis of a hierarchical model of mobile cloud computing.** Simulation Modelling Practice and Theory, 2014.



Availability: 0.9979091427

AverageCharge: 56.1680288982

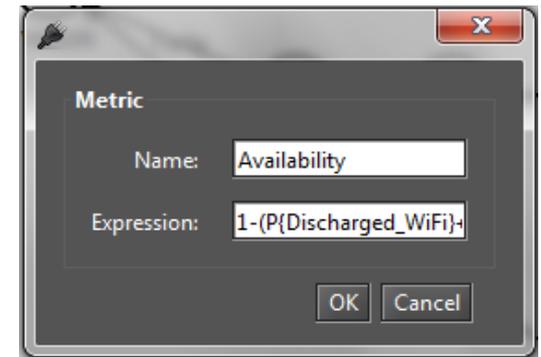
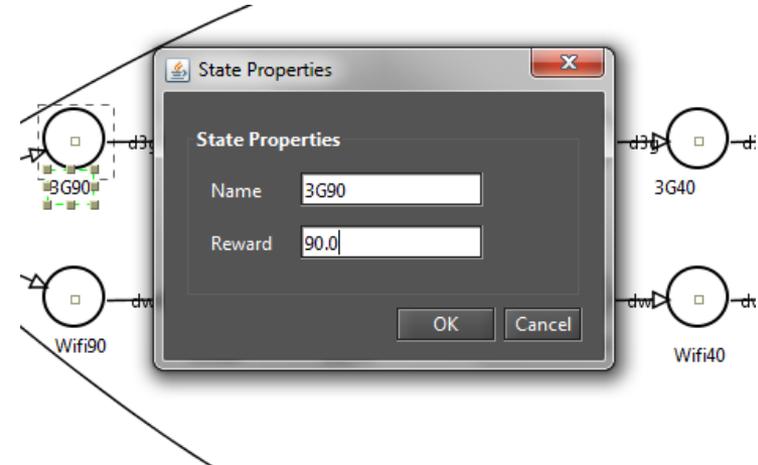
Métricas e rewards na CTMC



Availability: $1 - (P\{\text{Discharged_WiFi}\} + P\{\text{Discharged_3G}\})$
 AverageCharge: $R\{\}$

Métricas e rewards na CTMC

- ❑ **Rewards** podem ser atribuídos a cada **estado** da CTMC
- ❑ **Métricas** são **expressões** aritméticas simples, que podem incluir:
 - ❑ $P\{S_0\}$: Probabilidade de se estar num determinado estado S_0
 - ❑ $R\{S_0\}$: Taxa de reward atribuída a um estado S_0
 - ❑ $R\{\}$: Reward do sistema como um todo. Equivalente a $P\{S_0\} * R\{S_0\} + P\{S_1\} * R\{S_1\} + \dots$

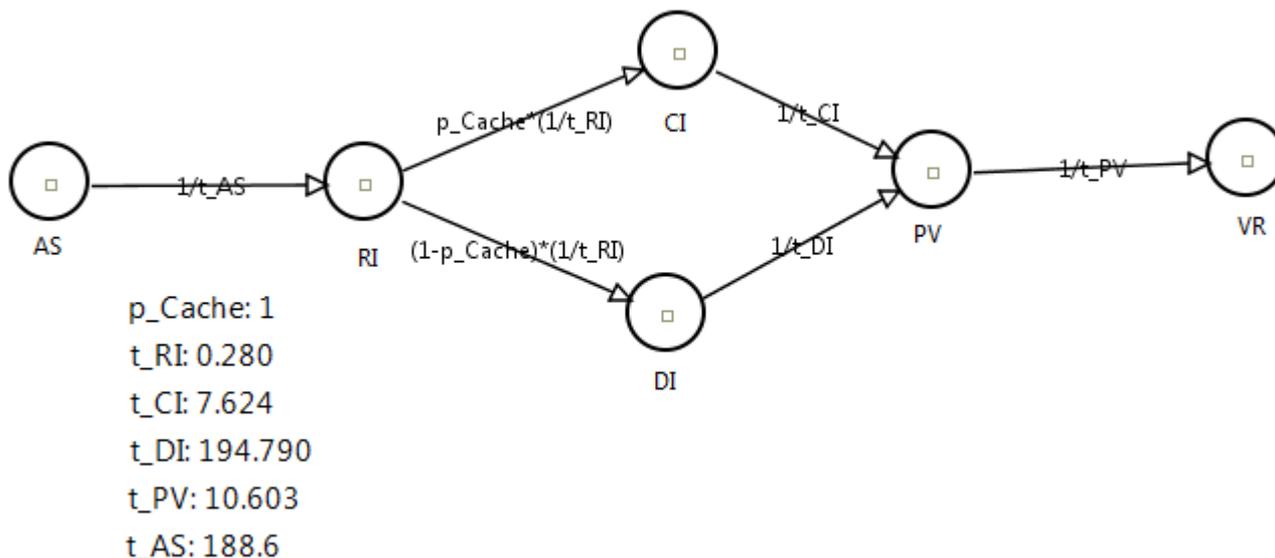


Availability: 0.9979091427

Tempo médio de absorção

Exemplo extraído de:

Eliomar G. Campos, Rubens Matos, Paulo R. M. Maciel, Francisco V. de Souza, Igor de O. Costa, Airtton Pereira. **Performance Evaluation of Virtual Machines Instantiation in a Private Cloud.** Aguardando revisão.





Tempo médio de absorção

Transient Analysis

Method:

Save CTMC Matrix Mean Time to Absorption (failure) Absorption Probability

Options

Time: Internal Step:

Precision:

Output: Point Curve



Tempo médio de absorção

Transient Analysis

Method:

Save CTMC Matrix Mean Time to Absorption (failure) Absorption Probability

Options

Time: Internal Step:

Precision:

Output: Point Curve

Analysis

Current Time: N. of iterations for a step:

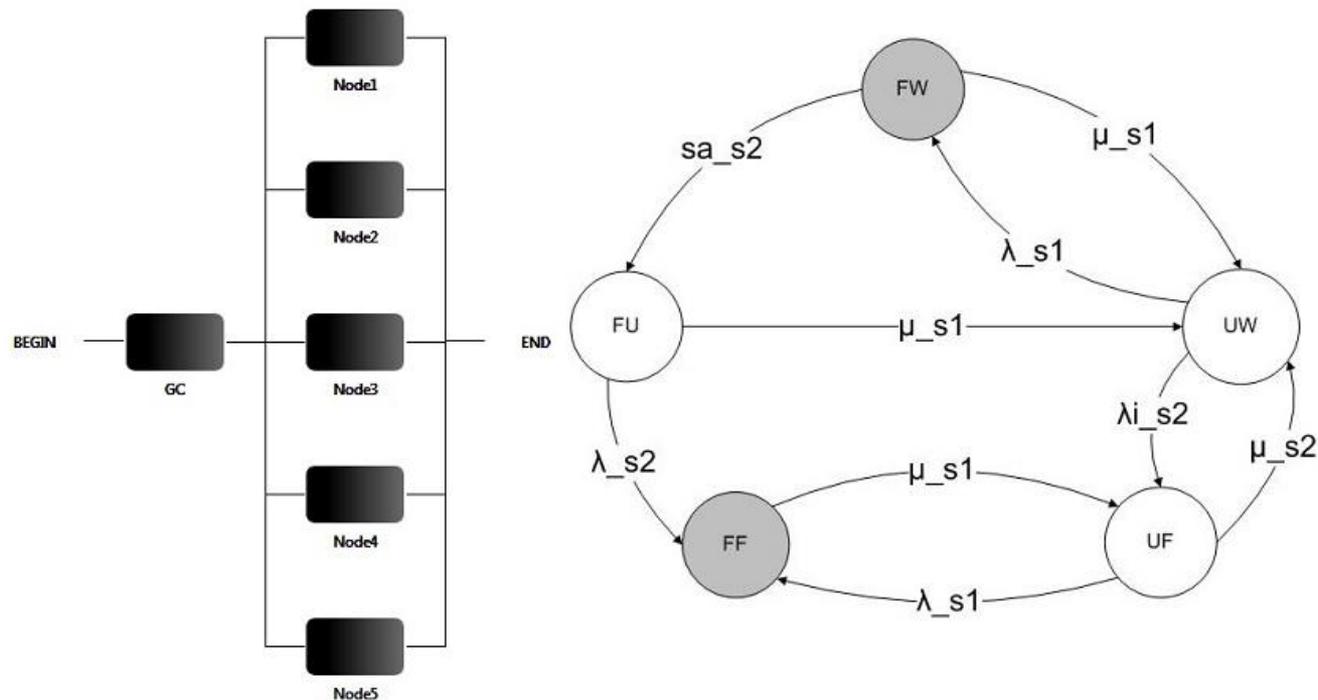
Results:

```
DI=0.0
PV=2.733E-4
VR=0.9997265999999999
AS=0.0
Absorption probability: 0.9997267
Mean Time to Absorption (MTTA): 18.506999999999998
```



Linguagem de Script – RBD e Markov Chain

- Estudo de caso extraído de: *Models for Dependability Analysis of Cloud. Computing Architectures for Eucalyptus. Platform.* J. Dantas, R. Matos, J. Araujo and P. Maciel





Modelo em cadeia de markov:

```
markov RedundantGC{
    state fu up;
    state fw;
    state ff;
    state uf up;
    state uw up;

    transition fw -> fu(rate = sa_s2);
    transition fu -> ff(rate = lambda_s2);
    transition ff -> uf(rate = mu_s1);
    transition uf -> uw(rate = mu_s2);
    transition uw -> fw(rate = lambda_s1);

    transition fw -> uw(rate = mu_s1);
    transition uw -> uf(rate = lambda_s2);
    transition uf -> ff(rate = lambda_s1);

    transition fw -> ff(rate= lambda_s2);
    transition fu -> uw(rate = mu_s1);

    metric aval = availability;
}
```



Modelos em RBD:

```
RBD NonRedundantGC{
    block hw(MTTF = mttfhw, MTTR = mttrhw);
    block so(MTTF = mttfso, MTTR = mttrso);
    block clc(MTTF = mttfclc, MTTR = mttrclc);
    block cc(MTTF = mttfcc, MTTR = mttrcc);
    block sc(MTTF = mttfsc, MTTR = mttrsc);
    block walrus(MTTF = mttfwalrus, MTTR = mttrwalrus);

    series s1(hw, so, clc, cc, sc, walrus);

    top s1;

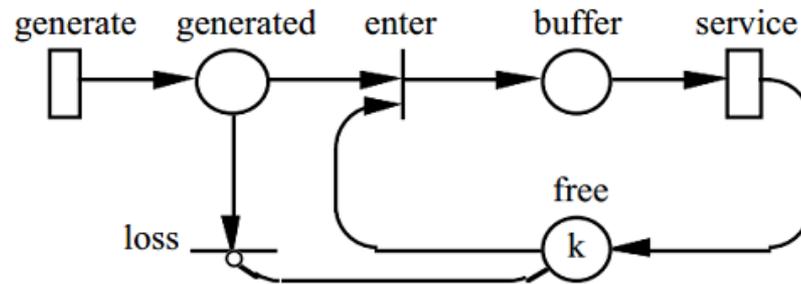
    metric aval = availability;
}

RBD Node{
    block hw(MTTF = mttfhw, MTTR = mttrhw);
    block so(MTTF = mttfso, MTTR = mttrso);
    block kvm(MTTF = mttfkvm, MTTR = mttrkvm);
    block nc(MTTF = mttfnc, MTTR = mttrnc);

    series b1(hw, so, kvm, nc);
```

Linguagem de Script - SPN

- Estudo de caso extraído de: *German, R. (1996), A concept for the modular description of stochastic petri nets , in 'Proc. 3rd Int. Workshop on Performability Modeling of Computer and Communication Systems'*





Modelo em SPN

```
SPN Foo{
  place gerados;
  place buffer;
  place livres(tokens = 10);

  timedTransition gerar(
    delay = 1,
    outputs = [gerados]
  );

  timedTransition servir(
    delay = servir,
    inputs = [buffer],
    outputs = [livres],
    serverType = "ExclusiveServer"
  );

  immediateTransition descarta(
    inputs = [gerados],
    inhibitors = [livres]
  );

  immediateTransition enfileira(
    inputs = [gerados, livres],
    outputs = [buffer]
  );

  metric m1 = stationaryProbability
    ( expression = "P{#buffer>0}" );
}
```



Avaliando os modelos:

```
main{
    lambda_s1 = 1/180.7212397;
    mu_s1 = 1/0.966902178;
    mu_s2 = 1/0.966902178;
    lambdai_s2 = 1/216.8654876049552;
    lambda_s2 = 1/180.7212397;
    sa_s2 = 1/0.005555555;

    (...)

    a = solve( model = NonRedundantGC, metric = aval );
    println( "Non Redundant GC Availability = " .. a );

    a2 = solve( model = RedundantGC, metric = aval );
    println( "Redundant GC Availability = " .. a2 );

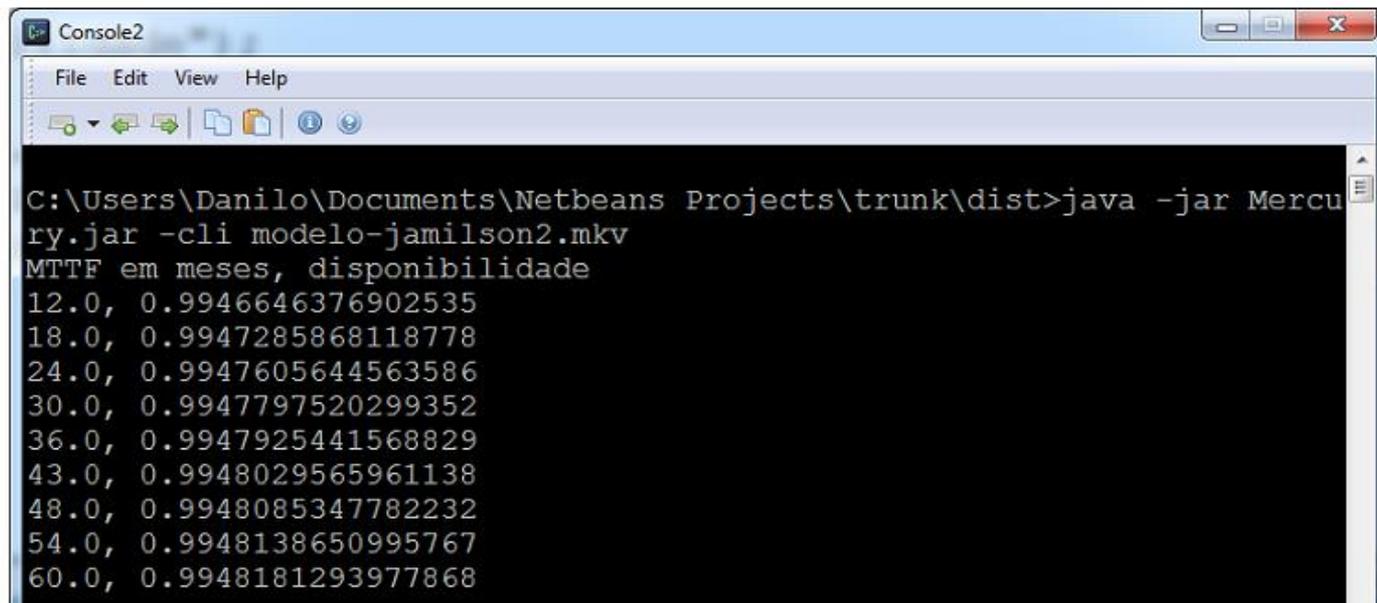
}
```

```
C:\Users\Danilo\Documents\Netbeans Projects\tru
ry.jar -cli modelo-jamilson.mkv
Non Redundant GC Availability = 0.9946672655743
Redundant GC Availability = 0.9999177939770505
C:\Users\Danilo\Documents\Netbeans Projects\tru
Ready
```



Realizando experimentos:

```
println("MTTF em meses, disponibilidade");  
  
for mttfmeses in [12, 18, 24, 30, 36, 43, 48, 54, 60]{  
    mttfhw = 30 * 24 * mttfmeses;  
    a = solve( model = NonRedundantCloud, metric = aval );  
    println(mttfmeses .. ", " .. a );  
}
```



```
Console2  
File Edit View Help  
C:\Users\Danilo\Documents\Netbeans Projects\trunk\dist>java -jar Mercury.jar -cli modelo-jamilson2.mkv  
MTTF em meses, disponibilidade  
12.0, 0.9946646376902535  
18.0, 0.9947285868118778  
24.0, 0.9947605644563586  
30.0, 0.9947797520299352  
36.0, 0.9947925441568829  
43.0, 0.9948029565961138  
48.0, 0.9948085347782232  
54.0, 0.9948138650995767  
60.0, 0.9948181293977868
```

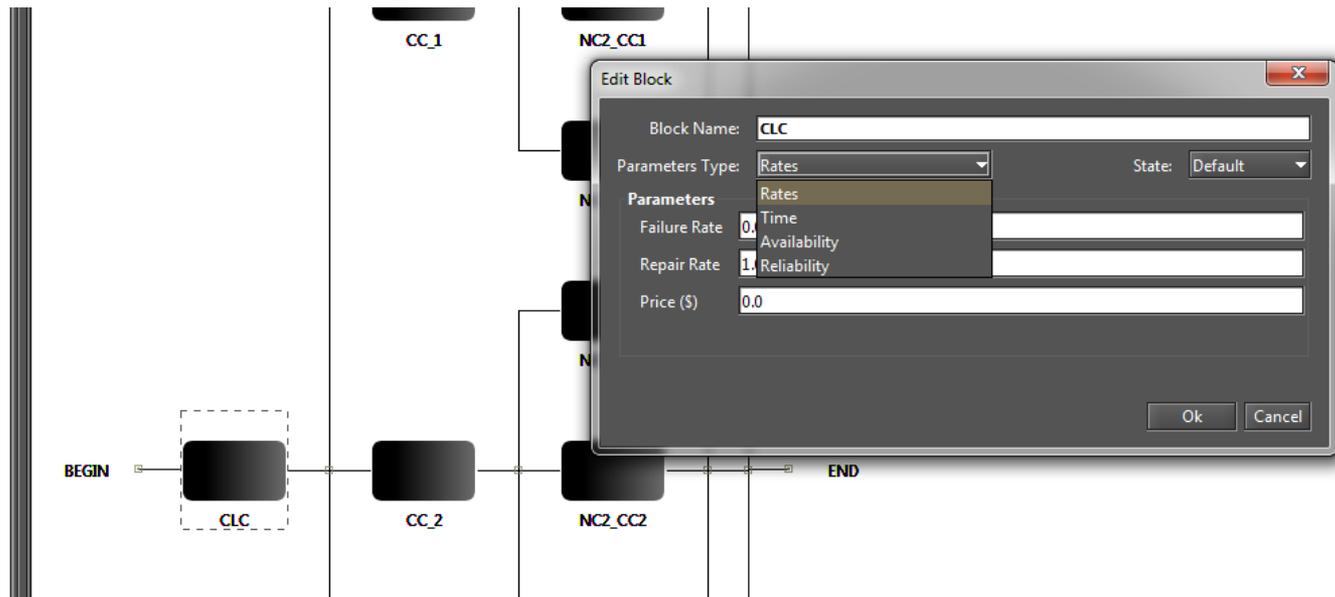


Melhorias na avaliação de RBD

- Possibilidade de configurar um valor fixo para a **confiabilidade** ou **disponibilidade** da cada **bloco**
- Cálculo de **downtime** e **uptime** anual

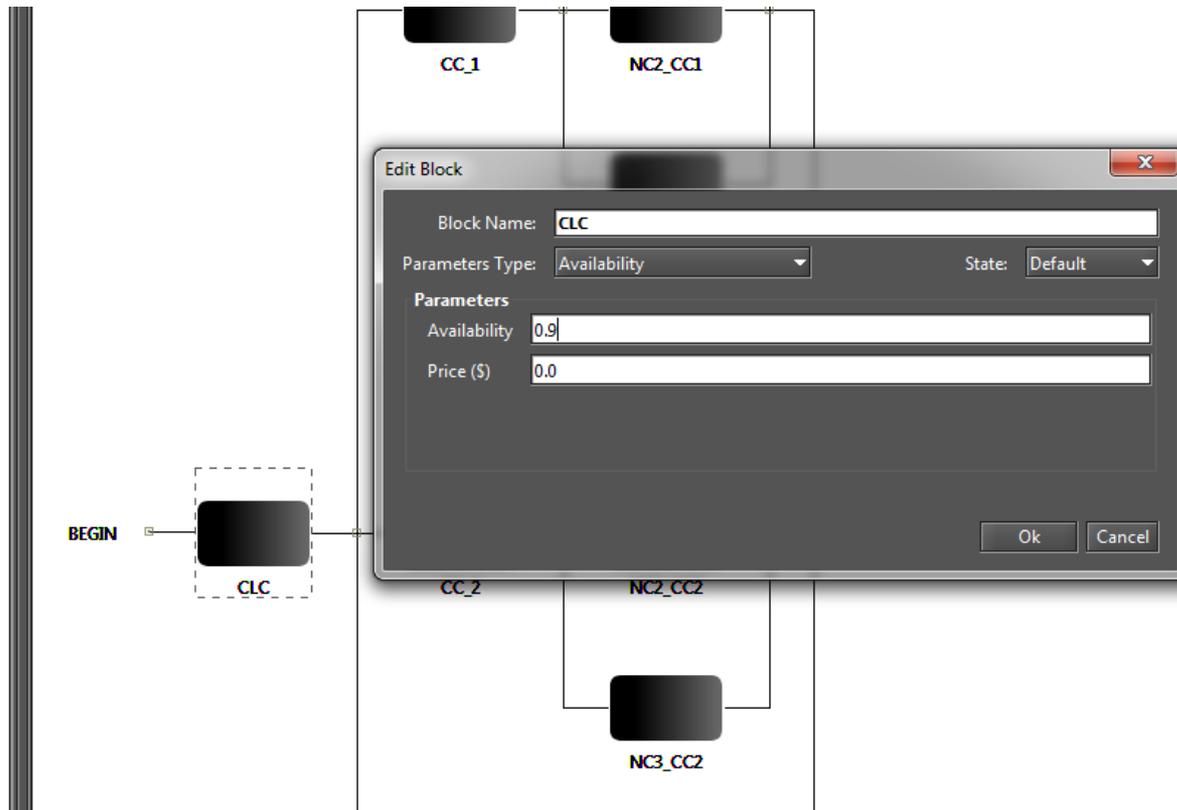


Melhorias na avaliação de RBD



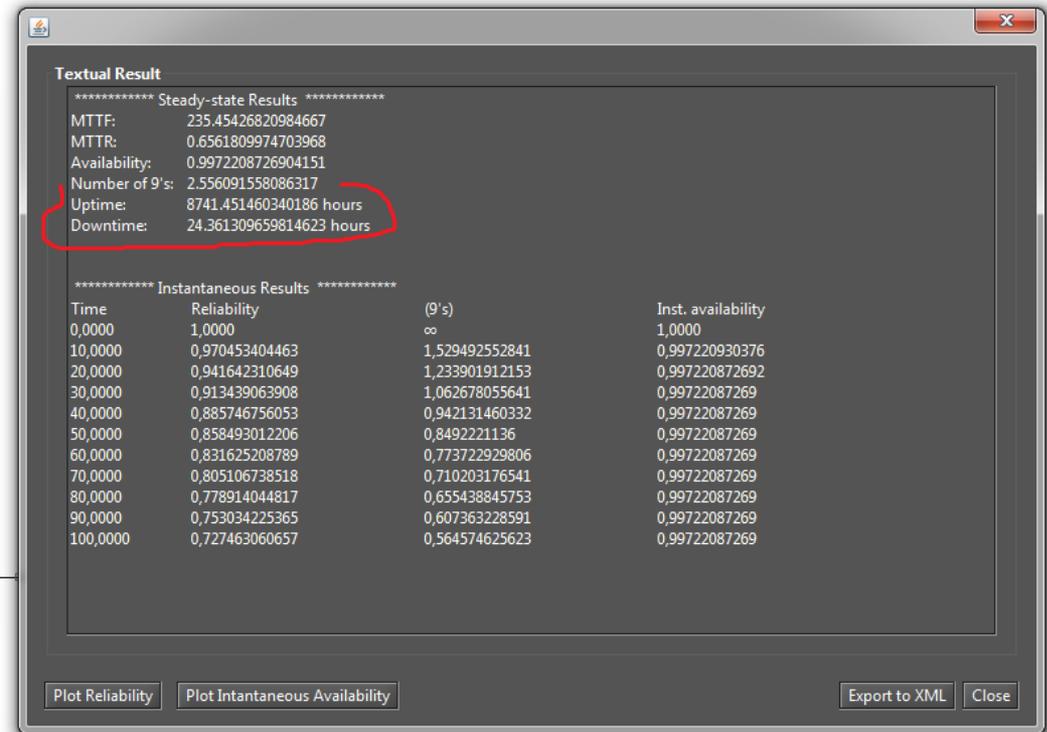
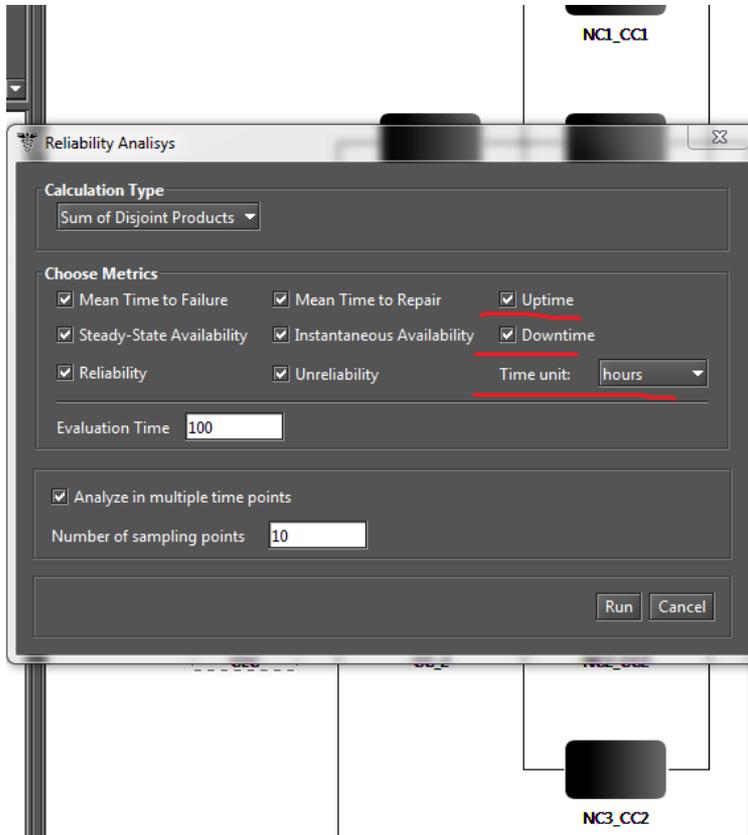


Melhorias na avaliação de RBD





Melhorias na avaliação de RBD



Geração Automática de Redes de Petri



- Utilização do Mercury como uma **biblioteca Java**
- Todas as funções do Mercury são **acessíveis** por meio da biblioteca.
- Pode-se criar ferramentas de alto nível para **geração/avaliação automática** de modelos.

Geração Automática de Redes de Petri



- Exemplo: GeoClouds

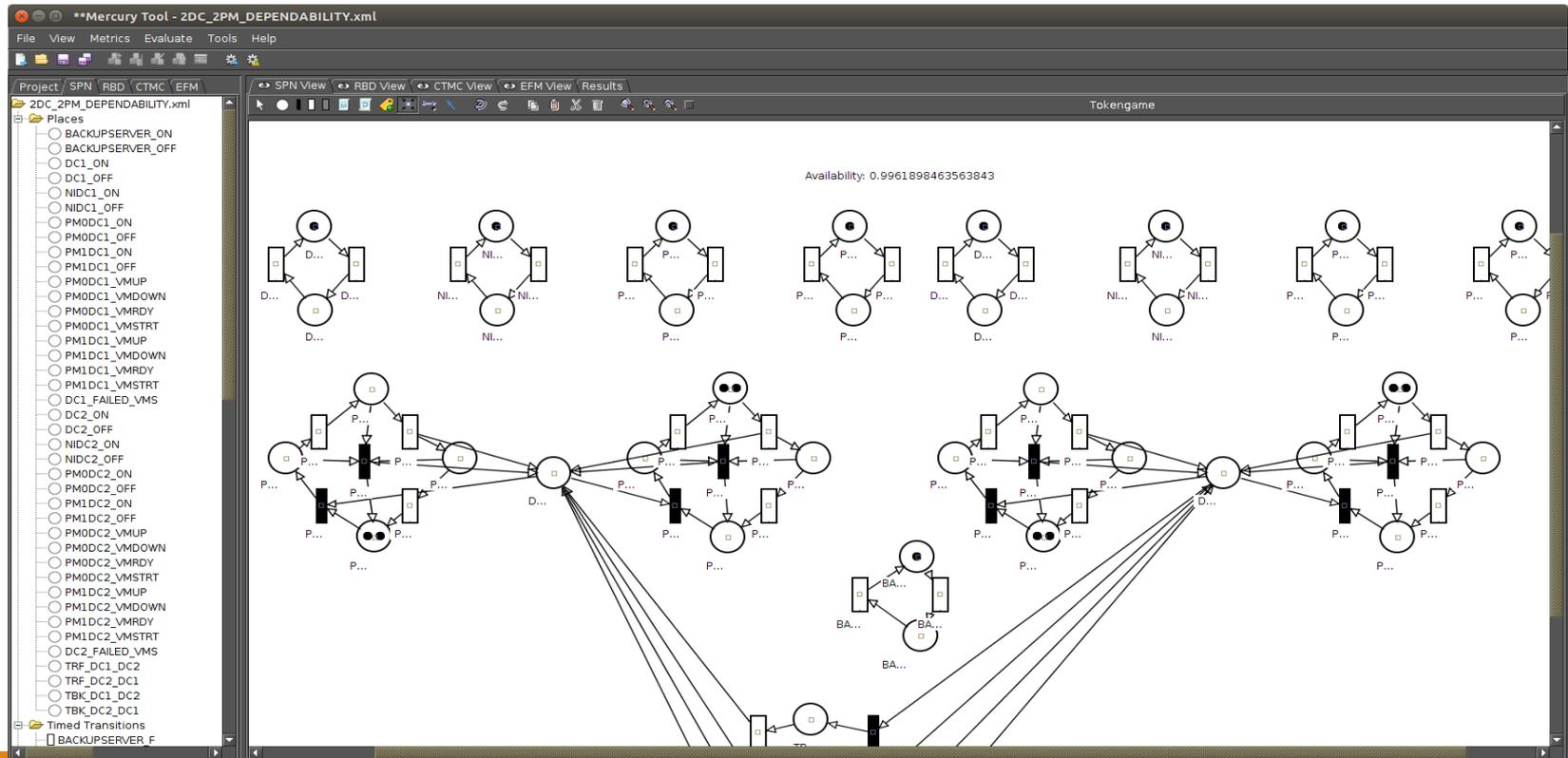
The screenshot shows the GeoClouds application interface. On the left, a 'Components' panel lists a 'Clouds system' containing a 'Backup Server', 'Load Parameters', and 'Time to Transfer VMs'. Under 'DC1', there is a list of 15 PM nodes (PM0DC1 to PM15DC1). The main area is a 'Playground' window displaying a map of Europe with three data centers marked: Paris, France; Amsterdam, The Netherlands; and Berlin, Germany. Below the map, there are tabs for 'Details', 'Evaluation Parameters', and 'Console Output'. The 'Evaluation Parameters' tab is active, showing a table with the following data:

Data Center #1	Data Center #2	MTT DC1 <=> DC2
DC1	DC2	2.0509228358899643

Geração Automática de Redes de Petri



- Exemplo: GeoClouds



Geração Automática de Redes de Petri



•Passos

- **Importar** o Mercury como uma biblioteca Java no projeto.
- Criar os **places** e **transições** (Análogo para o RBD e CTMC).
- Associar um gerador de **layout** (se necessário)
- **Avaliar** o modelo

Geração Automática de Redes de Petri



- Importar o Mercury como uma biblioteca Java no projeto.

```
Java - Eclipse SDK

Package Explorer
├── commons-math3-3.2.jar 37
├── commons-math3-3.2-test-sources.jar 37
├── commons-math3-3.2-tests.jar 37
├── commons-math3-3.2.jar 37
├── Blas.jar 96
├── cglib-nodep-2.2.jar 96
├── commons-io-2.4.jar 96
├── commons-lang3-3.2.jar 96
├── MomentMatching.jar 37
├── Mercury.jar 97
├── org-netbeans-modules-java-j2seproject-copylibstask.jar 11
├── poi-3.6-20091214.jar 11
├── poi-contrib-3.6-20091214.jar 11
├── poi-examples-3.6-20091214.jar 11
├── poi-ooxml-3.6-20091214.jar 11
├── poi-ooxml-schemas-3.6-20091214.jar 11
├── poi-scratchpad-3.6-20091214.jar 11
├── RandomVariateExperiments.jar 11
├── swing-layout-1.0.3.jar 11
├── swing-layout.jar 11
├── xml-writer-0.2.jar 11
├── xmlbeans-2.3.0.jar 11
├── jcommon-1.0.13.jar 11
├── jfreechart-1.0.10.jar 11
├── jgraph.jar 11
├── junit-3.8.1.jar 11
├── junit-3.8.2.jar 11
├── junit-4.5.jar 11
├── log4j-1.2.13.jar 11
├── looks-2.3.0.jar 11

VmBehaviorLayout
Evaluator.java
JFrameDesdac

package org.modcs.geoclouds.stochasticmodel;

import java.util.ArrayList;

public class Evaluator {

    // get UISynchronize injected as field
    @Inject
    UISynchronize sync;
    ArrayList<ResultsStationary> rs;
    Job job1, job2;
    boolean isEvaluating;

    /**
     * @return the isEvaluating
     */
    public synchronized boolean isEvaluating() {
        return isEvaluating;
    }

    /**
     * @param isEvaluating the isEvaluating to set
     */
    public synchronized void setEvaluating(boolean
        this.isEvaluating = isEvaluating;
    }

    /**
     * print string
     */
    private void printString(final String string) {
        // If you want to update the UI
        sync.asyncExec(new Runnable() {
            @Override
            public void run() {
                System.out.print(string);
            }
        });
    }
}
```

Geração Automática de Redes de Petri



- Criar os places e transições

```
public SimpleComponentSPN(BasicComponent basic) throws Exception {  
  
    String name = basic.getName().replaceAll("\\s", "").toUpperCase();  
  
    upPlace = new Place(name + "_ON", 1);  
    downPlace = new Place(name + "_OFF", 0);  
  
    places.add(upPlace);  
    places.add(downPlace);  
  
    failPlace = new TransitionExponential(name + "_F", basic.getMttf());  
    failPlace.setArcInputs(new ArrayList<ArcInput>());  
    failPlace.getArcInputs().add(new ArcInput(upPlace, 1));  
  
    failPlace.setArcOutputs(new ArrayList<ArcOutput>());  
    failPlace.getArcOutputs().add(new ArcOutput(downPlace, 1));  
  
    repairPlace = new TransitionExponential(name + "_R", basic.getMttr());  
    repairPlace.setArcInputs(new ArrayList<ArcInput>());  
    repairPlace.getArcInputs().add(new ArcInput(downPlace, 1));  
  
    repairPlace.setArcOutputs(new ArrayList<ArcOutput>());  
    repairPlace.getArcOutputs().add(new ArcOutput(upPlace, 1));  
  
    transitions.add(failPlace);  
    transitions.add(repairPlace);  
}
```

Geração Automática de Redes de Petri



- Associar o Layout

```
@Override
public void updateLayout(Point2D inserctionPoint) {
    this.insertPositionPlace(simpleComponentSPN.getUpPlace(), inserctionPoint, 0, 0);
    this.insertPositionPlace(simpleComponentSPN.getDownPlace(), inserctionPoint, 0, 100);

    this.insertPositionTransition(simpleComponentSPN.getFailTransition(), inserctionPoint, 40, 10);
    this.insertPositionTransition(simpleComponentSPN.getRepairTransition(), inserctionPoint, -50, 10);
}
```

Geração Automática de Redes de Petri



- Avaliar o Modelo

```
EDSPN spn = null;
try {
    spn = Parser.getInstance().parse(hlm);
} catch (Exception e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
    return Status.CANCEL_STATUS;
}

println("SPN created ...\n");

LayoutManager.getInstance().updateLayout(
    Parser.getInstance().getCloudComponentSPN());
println("Graphical Layout Updated...\n");

println("Evaluating ... \nThe process may take several minutes...\n");
setEvaluating(true);

rs = JFrameDesdacTool.stationaryAnalisys(spn, hlm.getStationaryParameters());
```



New Feature – Propriedades estruturais

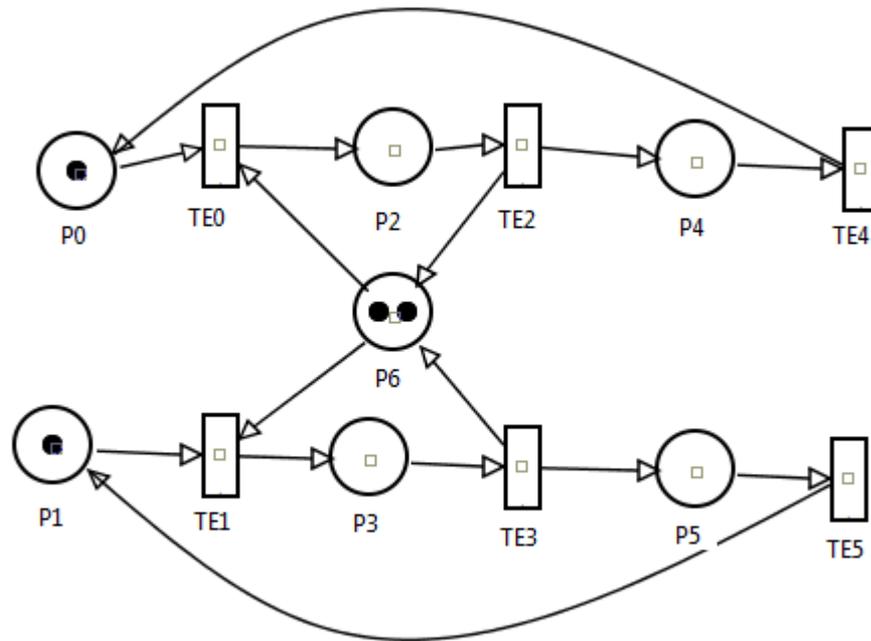
- Método: Destrinchar a Petri net como a combinação de diferentes sistemas de inequações.
- Resultados : Propriedades inerentes da estrutura
 - Ex : boundedness, conservativeness, repetitiveness and consistency



New Feature – Propriedades estruturais

Estudo de caso – Communicating System Modeling

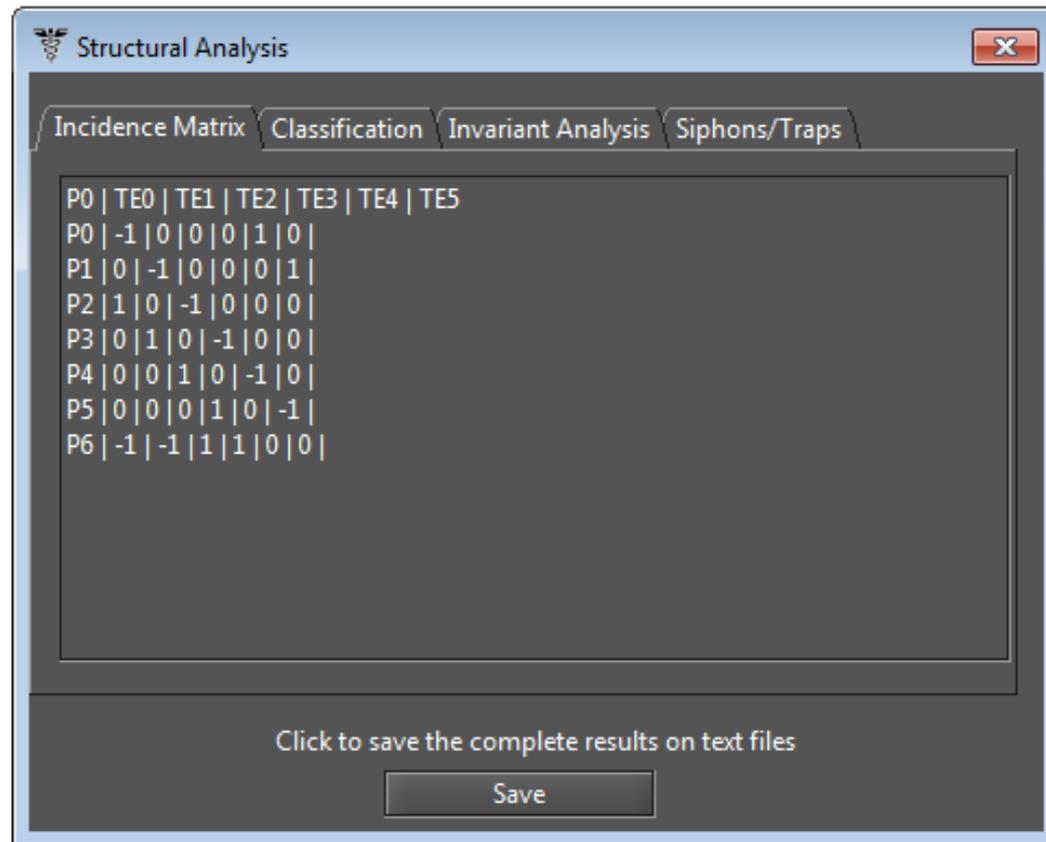
Link: <http://www.modcs.org/wp-content/uploads/2008/09/pnposgrad2.pdf>





New Feature – Propriedades estruturais

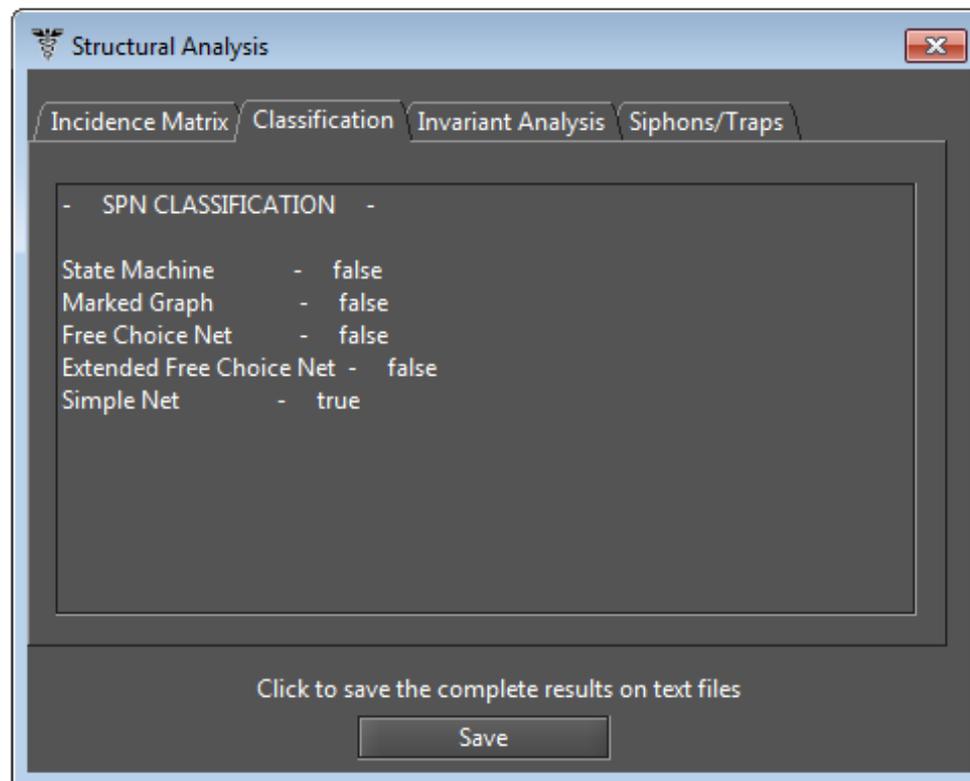
- 1° - Teoria Matricial
 - Matriz de entrada(I), saída(O) e incidência(C)





New Feature – Propriedades estruturais

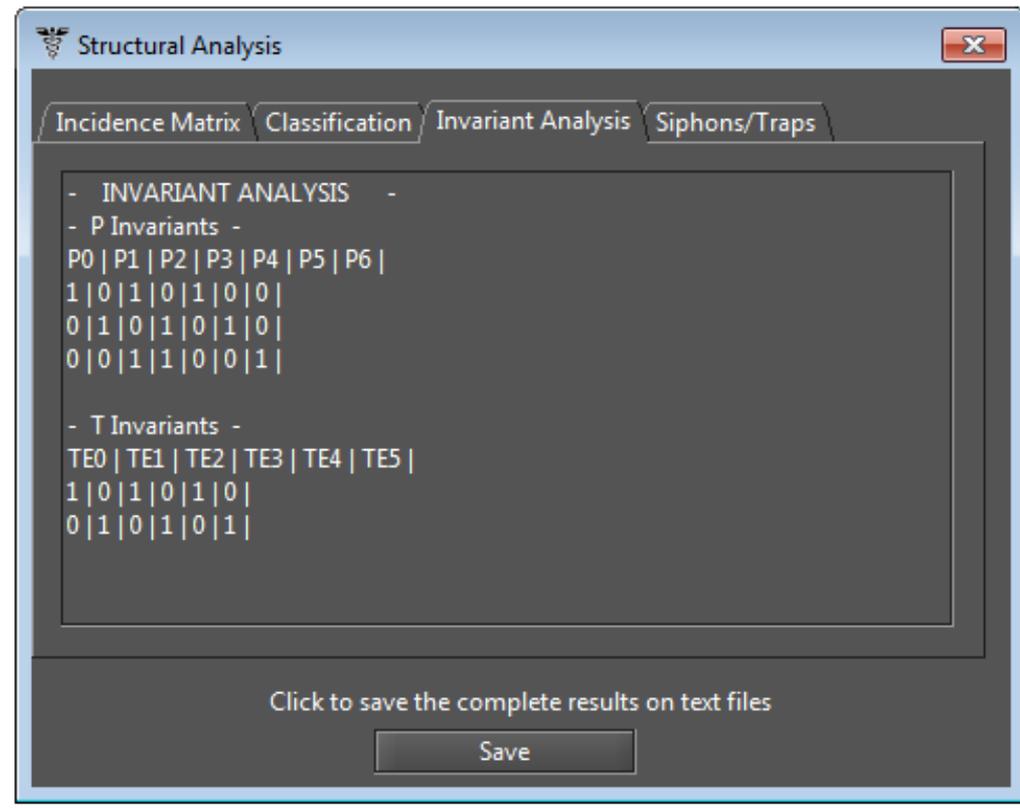
- 2° - Sub-Classes
 - Petri Net, Simple Net, Extended Free Choice Net, Free Choice Net, Market Graph, State Machine





New Feature – Propriedades estruturais

- 3° - Invariantes de Lugar e Transição
 - Usa o algoritmo de computação sobre matriz identidade





New Feature – Propriedades estruturais

- 4° - Siphon/Traps
 - Usa as mínimas combinações de linhas em (C) para o cálculo

