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

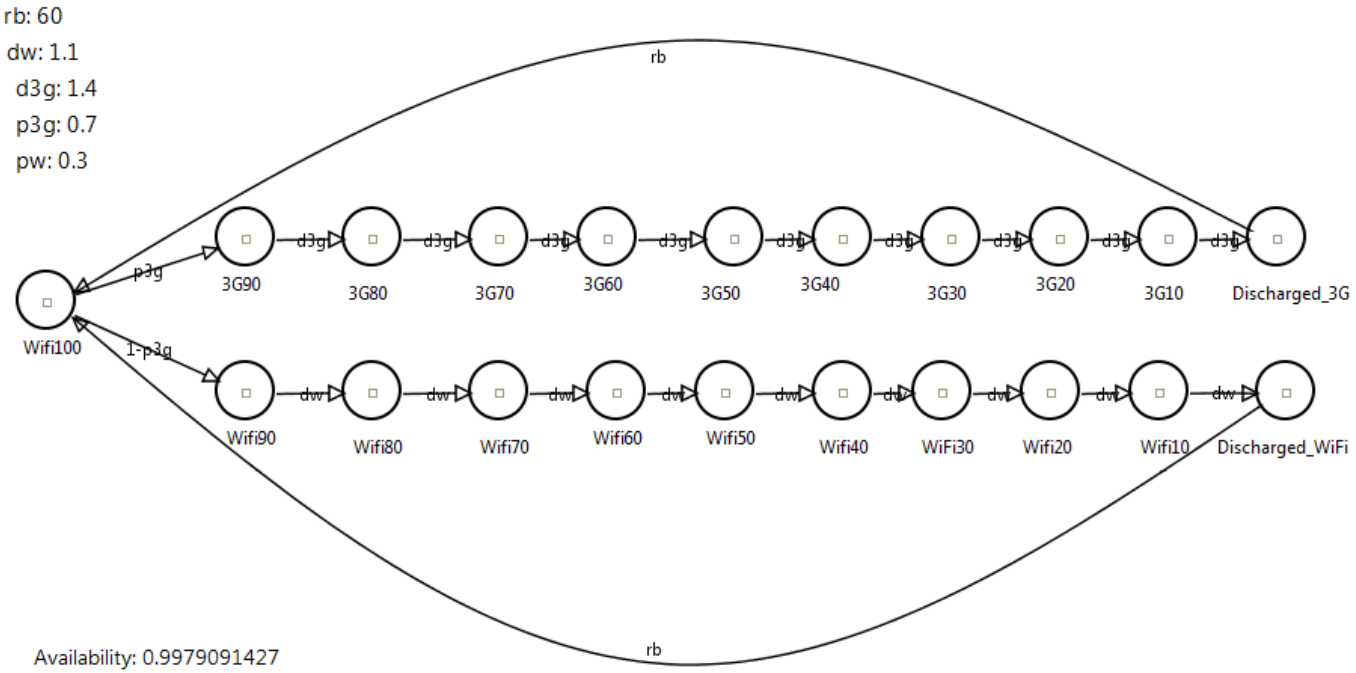
Ajustes da Ferramenta

- ✓ 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

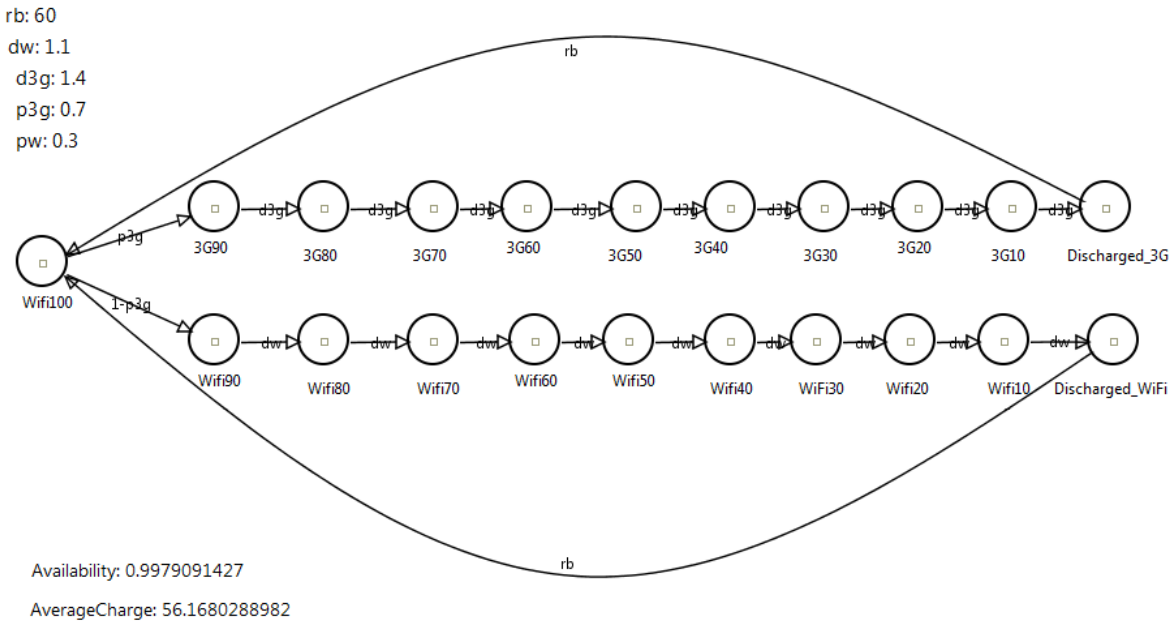
Métricas e rewards na CTMC

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.



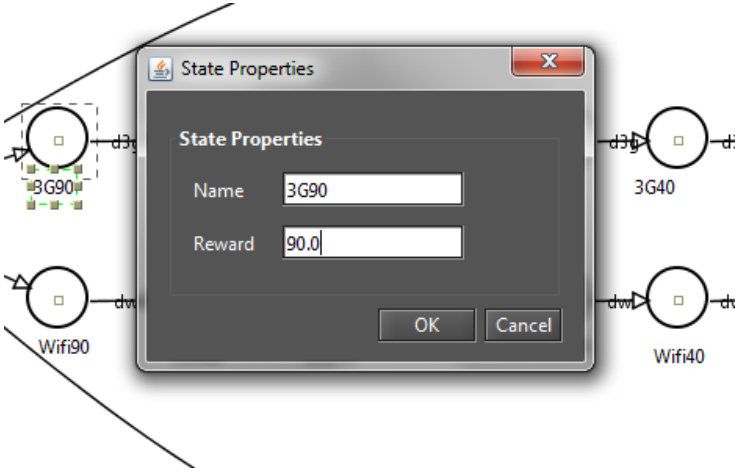
Métricas e rewards na CTMC

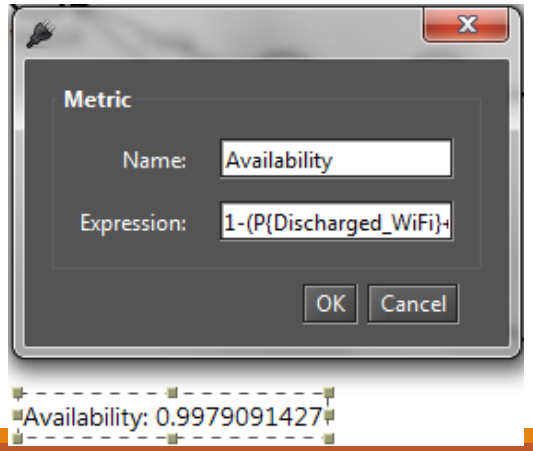


Availability: 1-(P{Discharged_WiFi}+P{Discharged_3G})
AverageCharge: R{}

Métricas e rewards na CTMC

- Rewards podem ser atribuídos a cada estado da CTMC
- Métricas são expressões aritméticas simples, que podem incluir:
  - P{S0}: Probabilidade de se estar num determinado estado S0
  - R{S0}: Taxa de reward atribuída a um estado S0
  - R{}: Reward do sistema como um todo. Equivalente a P{S0}*R{S0}+P{S1}*R{S1}+ ...

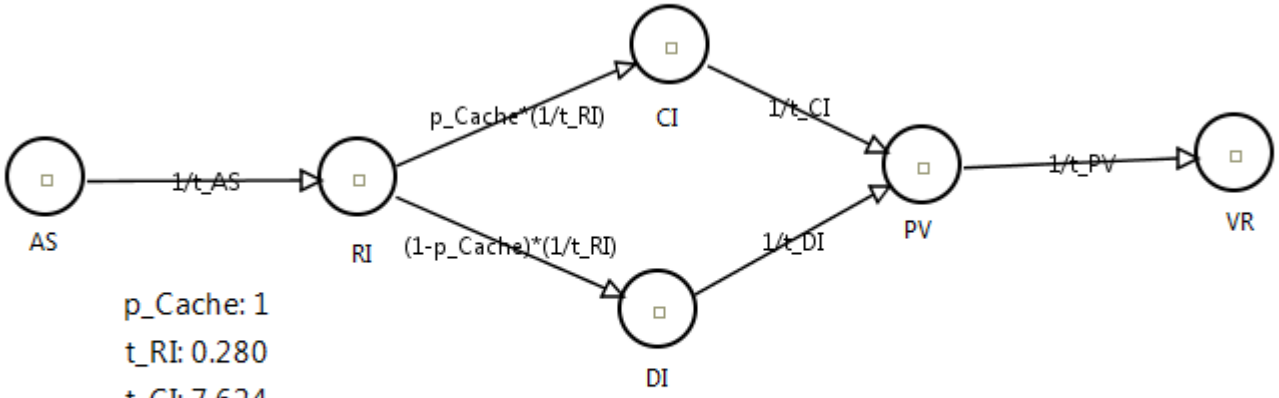




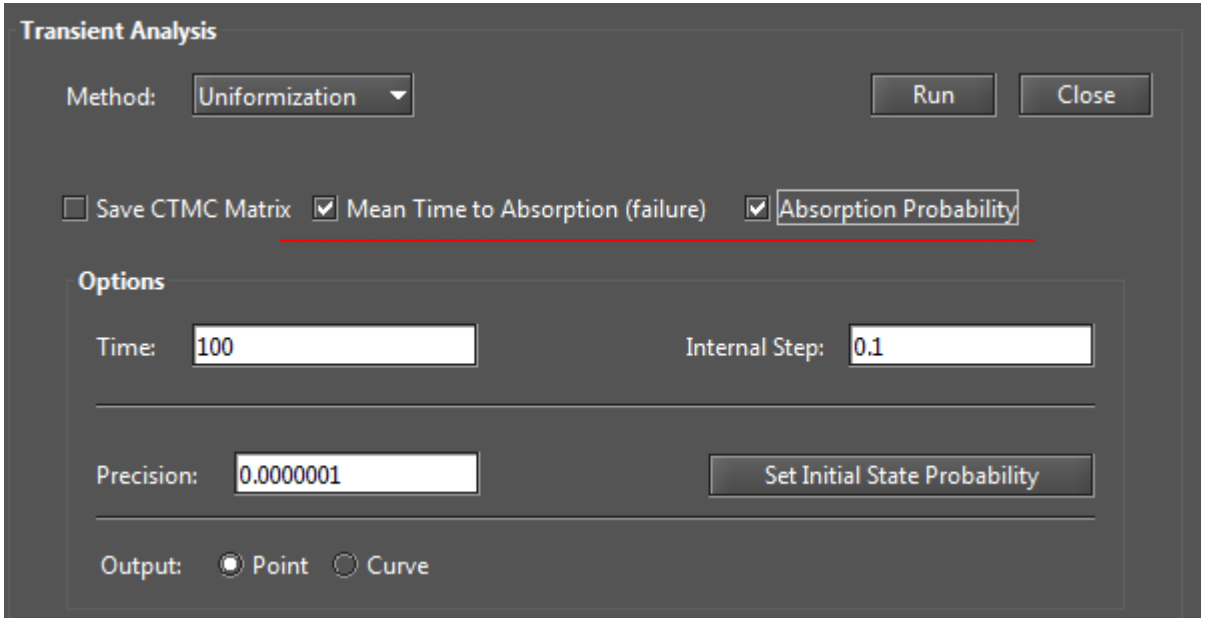
Tempo médio de absorção

Exemplo extraído de:

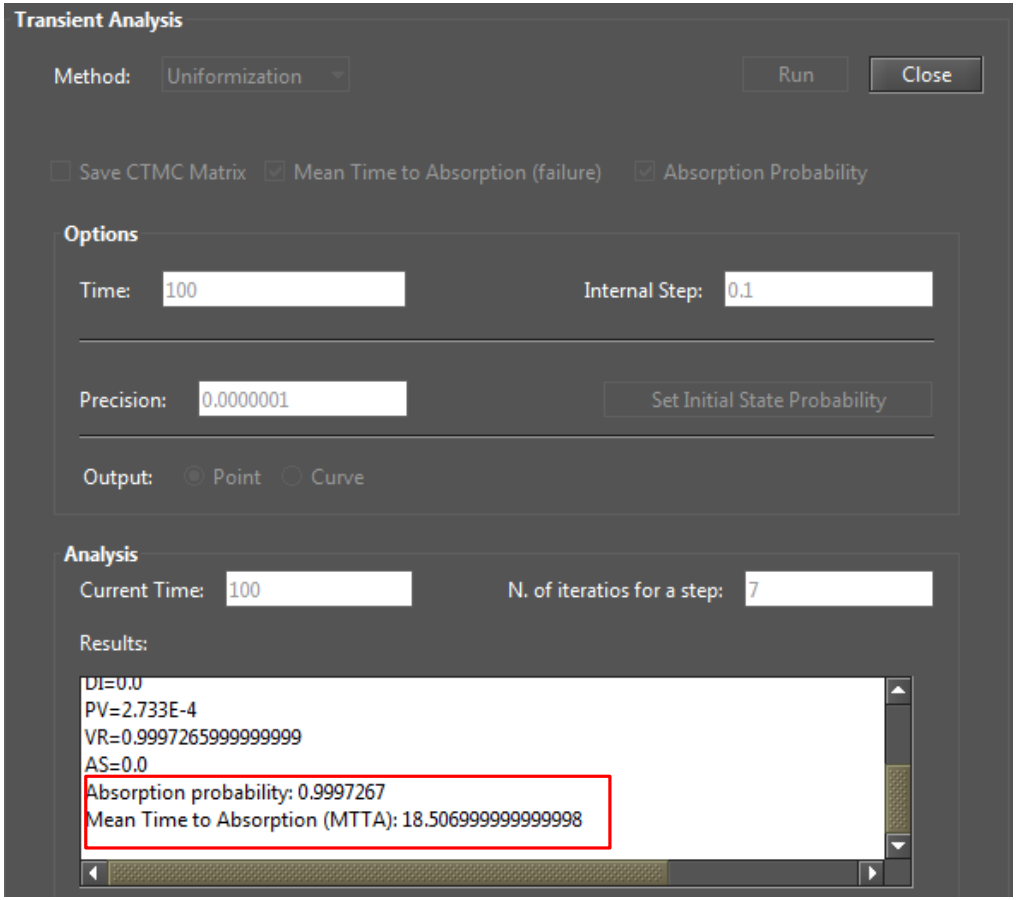
Eliomar G. Campos, Rubens Matos, Paulo R. M. Maciel, Francisco V. de Souza, Igor de O. Costa, Airton Pereira. **Performance Evaluation of Virtual Machines Instantiation in a Private Cloud.** Aguardando revisão.



Tempo médio de absorção

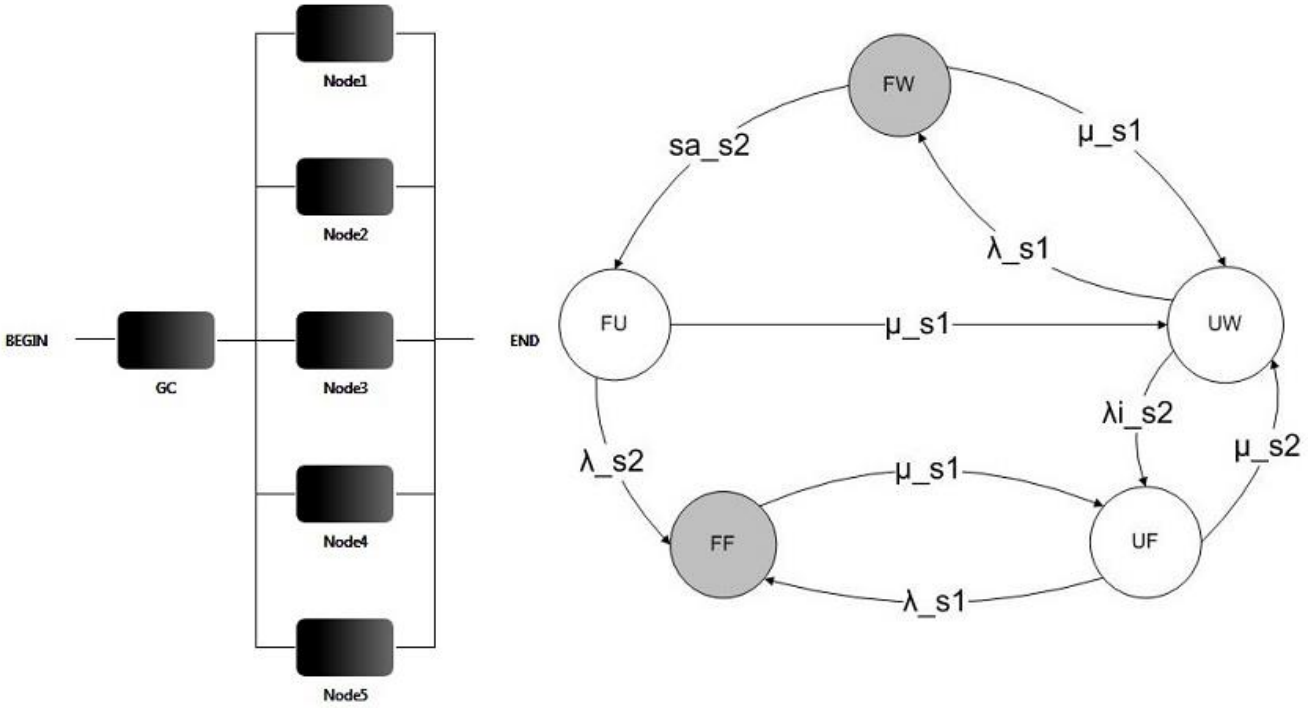


Tempo médio de absorção



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:

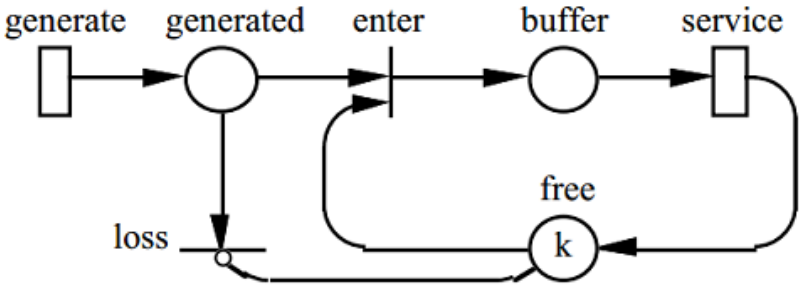
```
markov RedundantGC{
    state fu up;
    state fw;
    state ff;
    state uf up;
    state uw up;

    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 = lambdai_s2);
    transition uf -> ff(rate = lambda_s1);

    transition fw -> ff(rate= lambdai_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'*

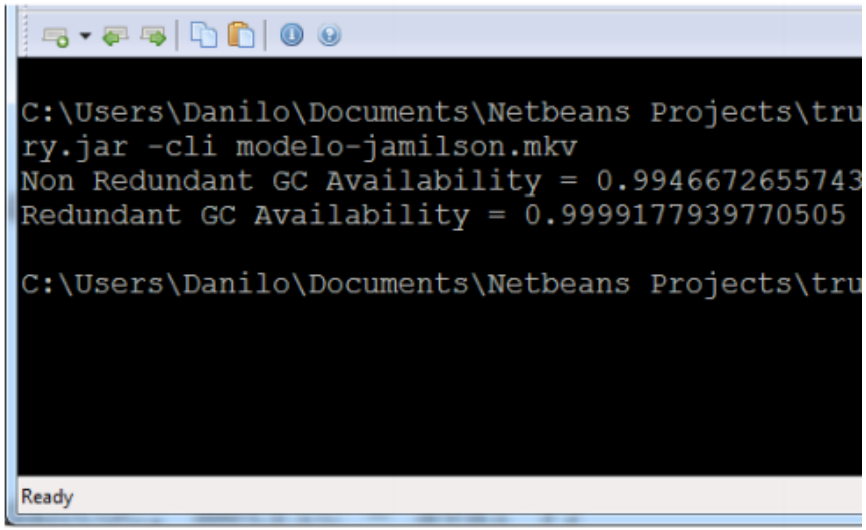


Modelo em SPN

```
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}" );
}
```

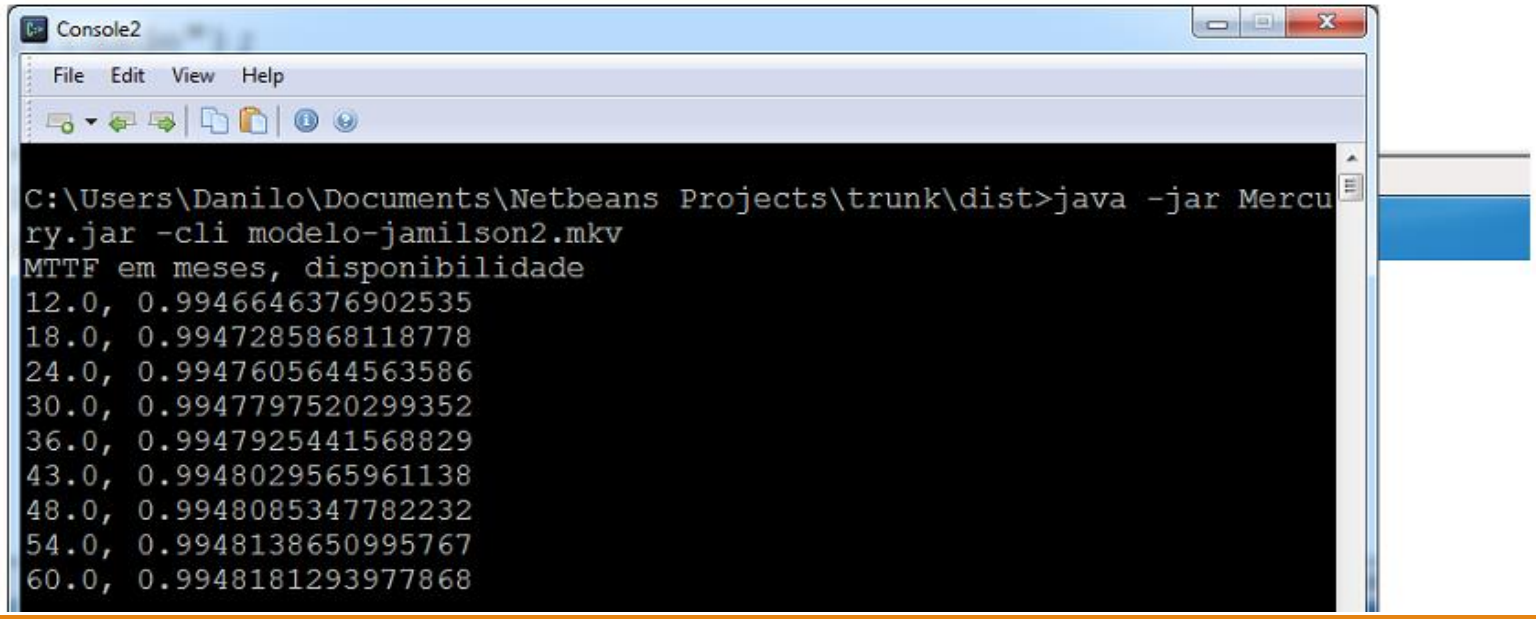
Avaliando os modelos:

```
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 );

}
```



Realizando experimentos:

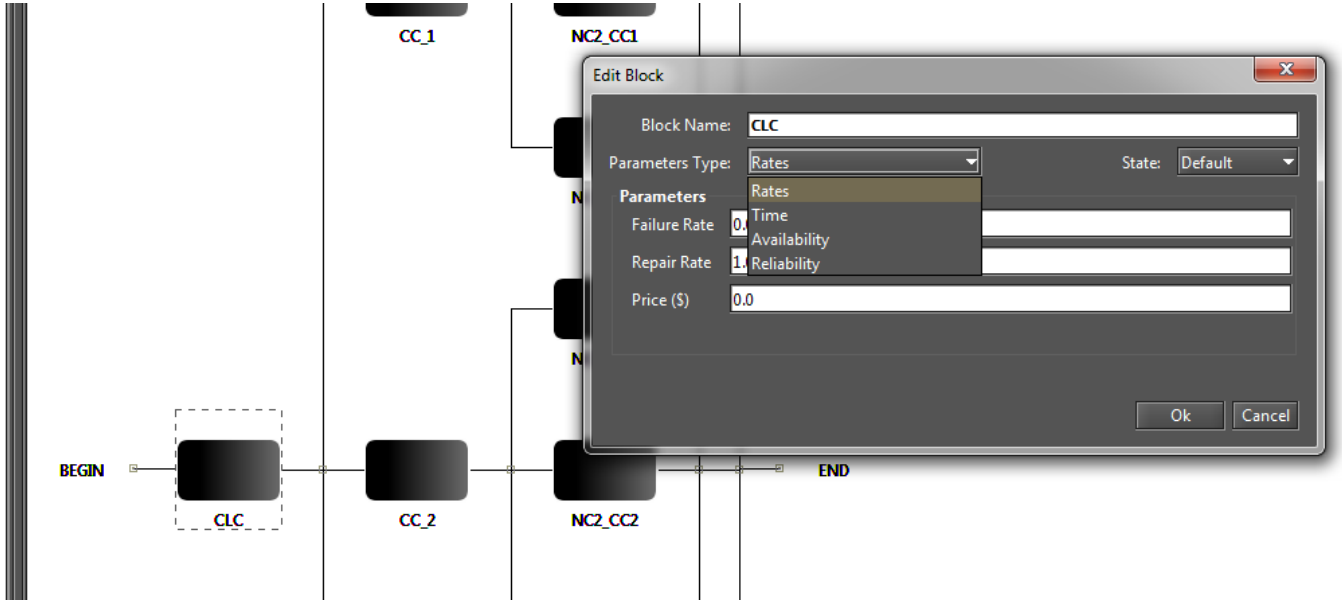
```
println("MTTF em meses, disponibilidade");

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 );
}
```



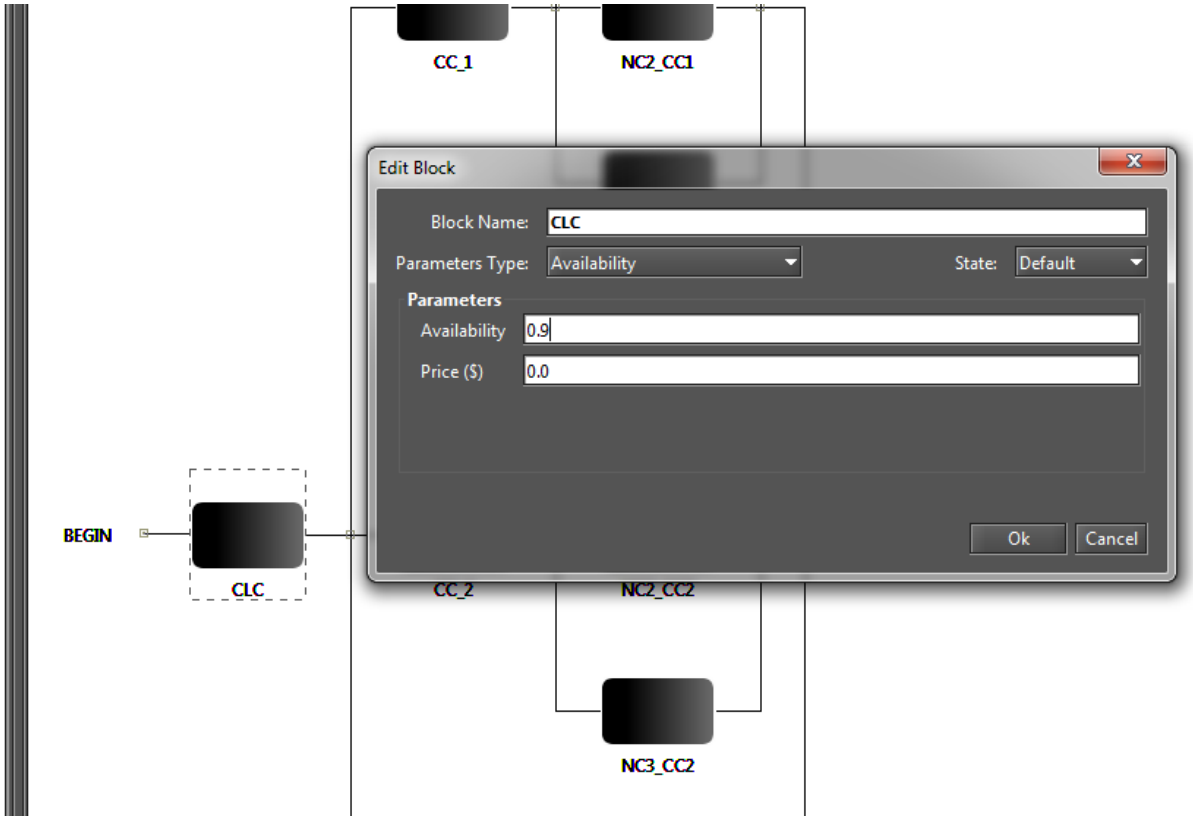
Melhorias na avaliação de RBD

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

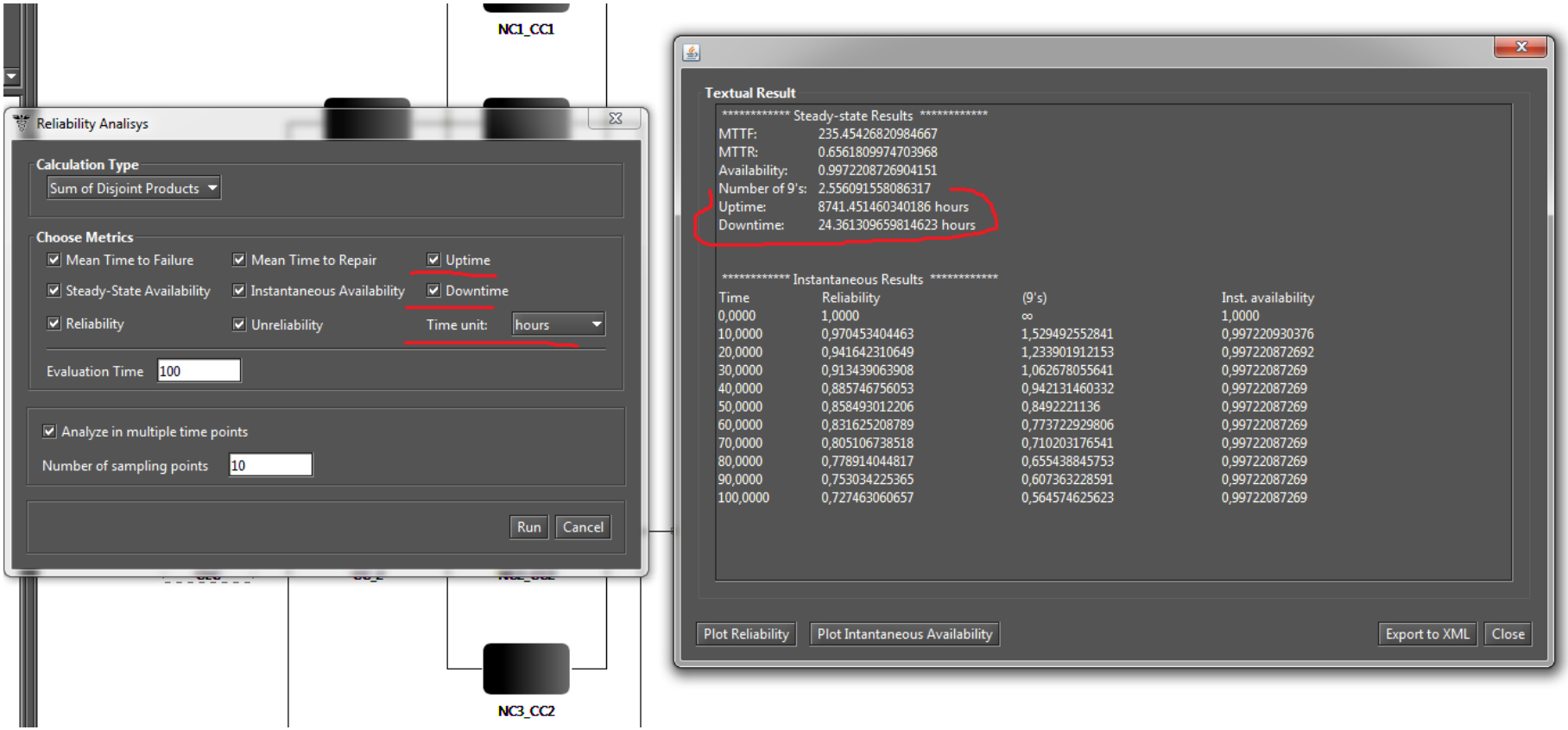
Melhorias na avaliação de RBD



Melhorias na avaliação de RBD



Melhorias na avaliação de RBD

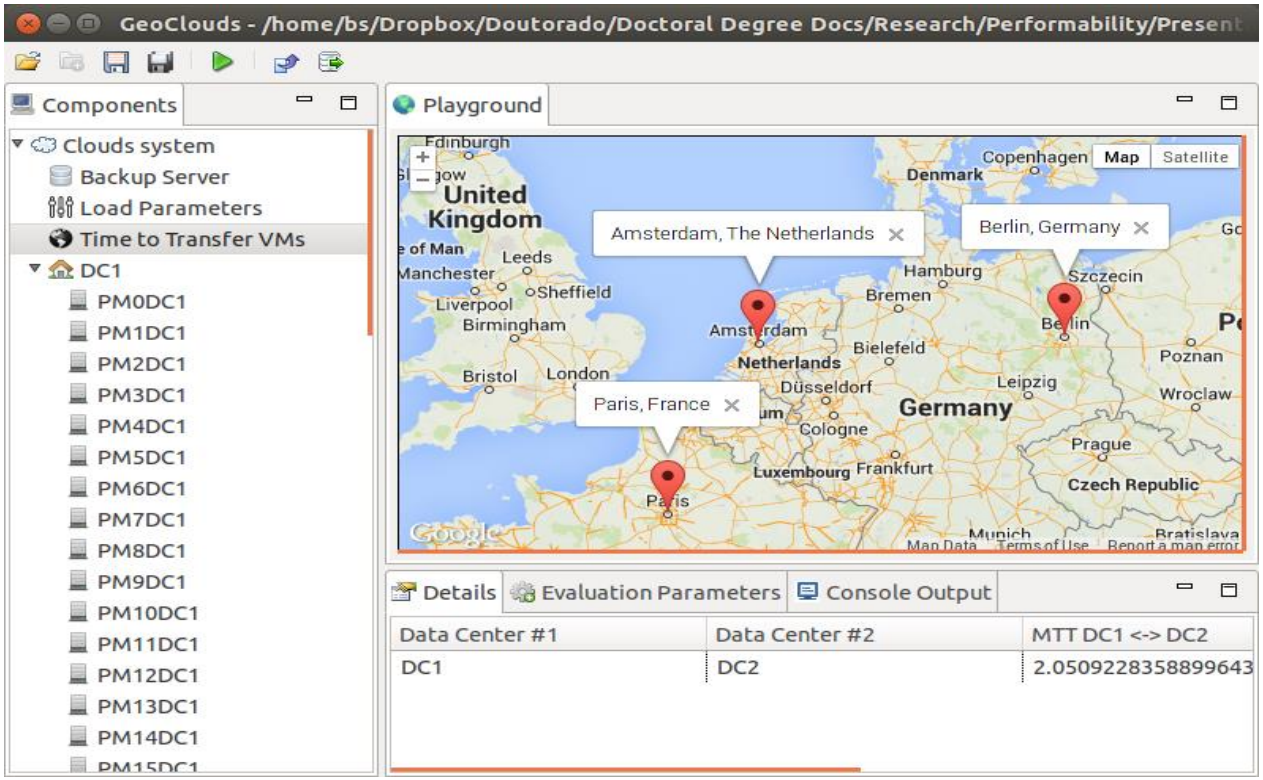


Geração Automática de Redes de Petri

- 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.

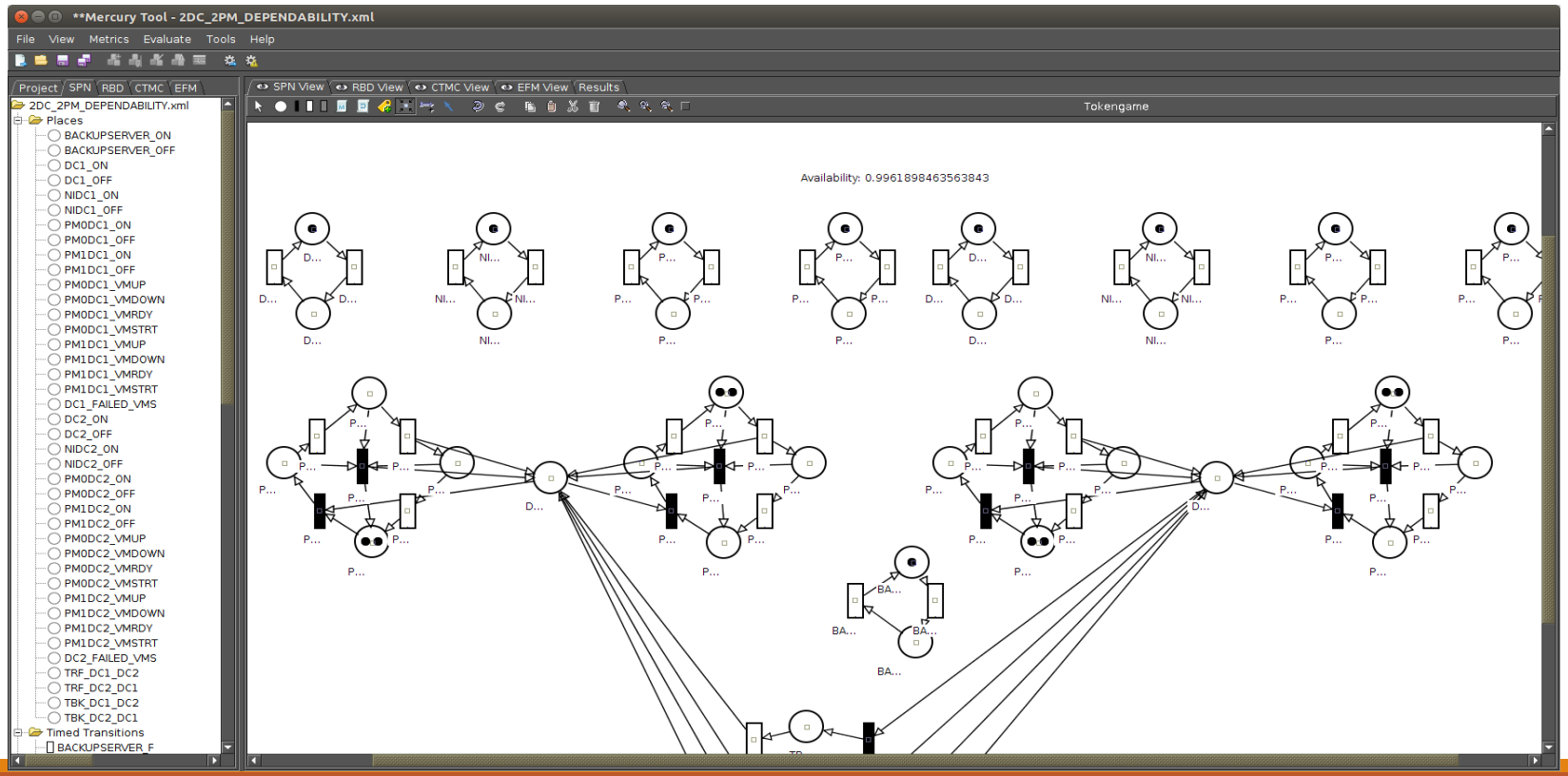
Geração Automática de Redes de Petri

- Exemplo: GeoClouds



Geração Automática de Redes de Petri

- Exemplo: GeoClouds

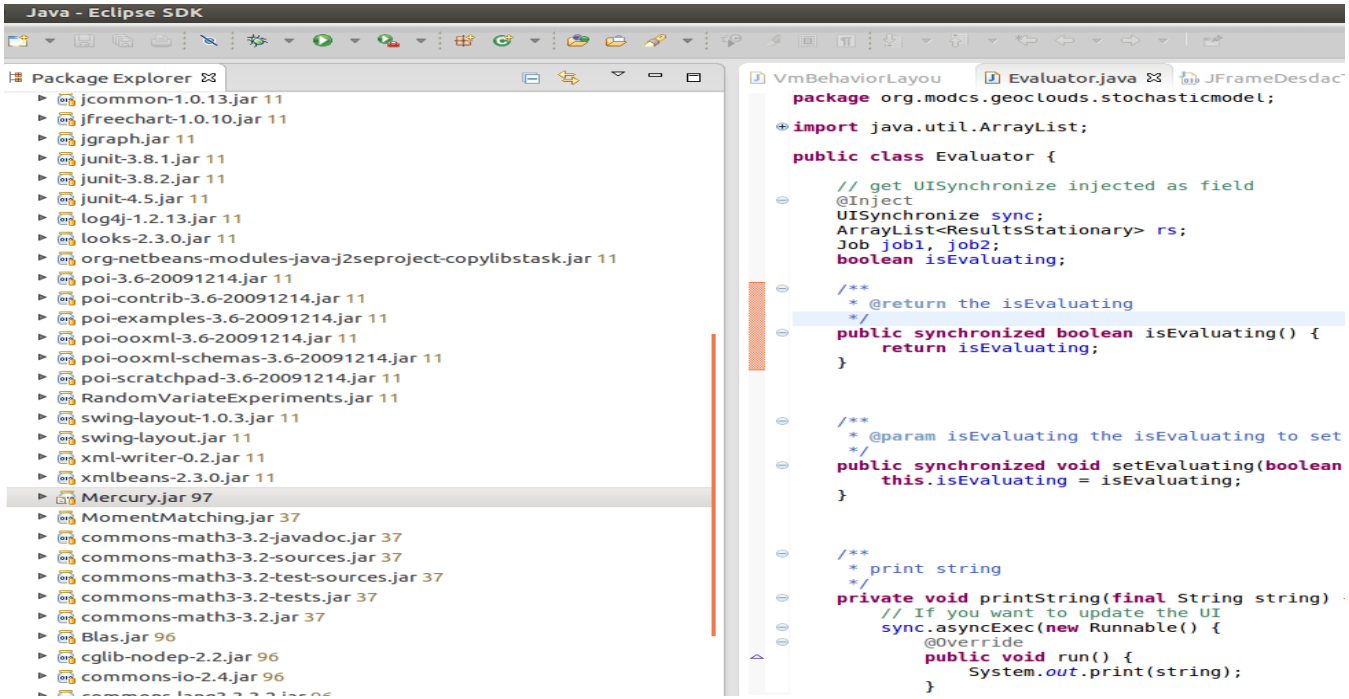


Geração Automática de Redes de Petri

- 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.



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;
}

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

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

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

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

New Feature – Propriedades estruturais

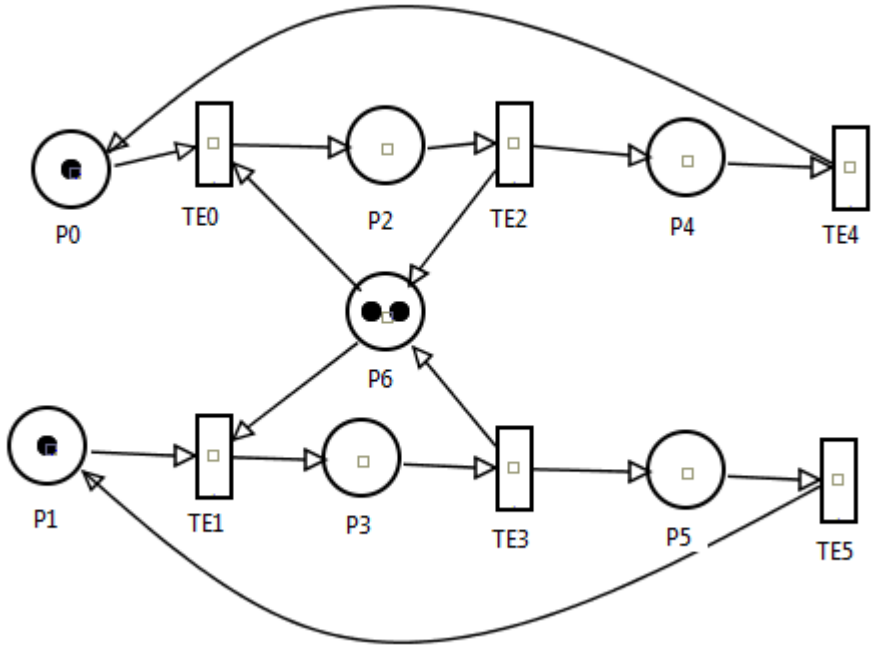
- Método: Destrinchar a Petri net como a combinação de diferentes sistemas de inequações.

- Resultados : Propriedades inerentes da estrutura
  - Ex : boundedness, conservativeness, repetitiveness and consistency

New Feature – Propriedades estruturais

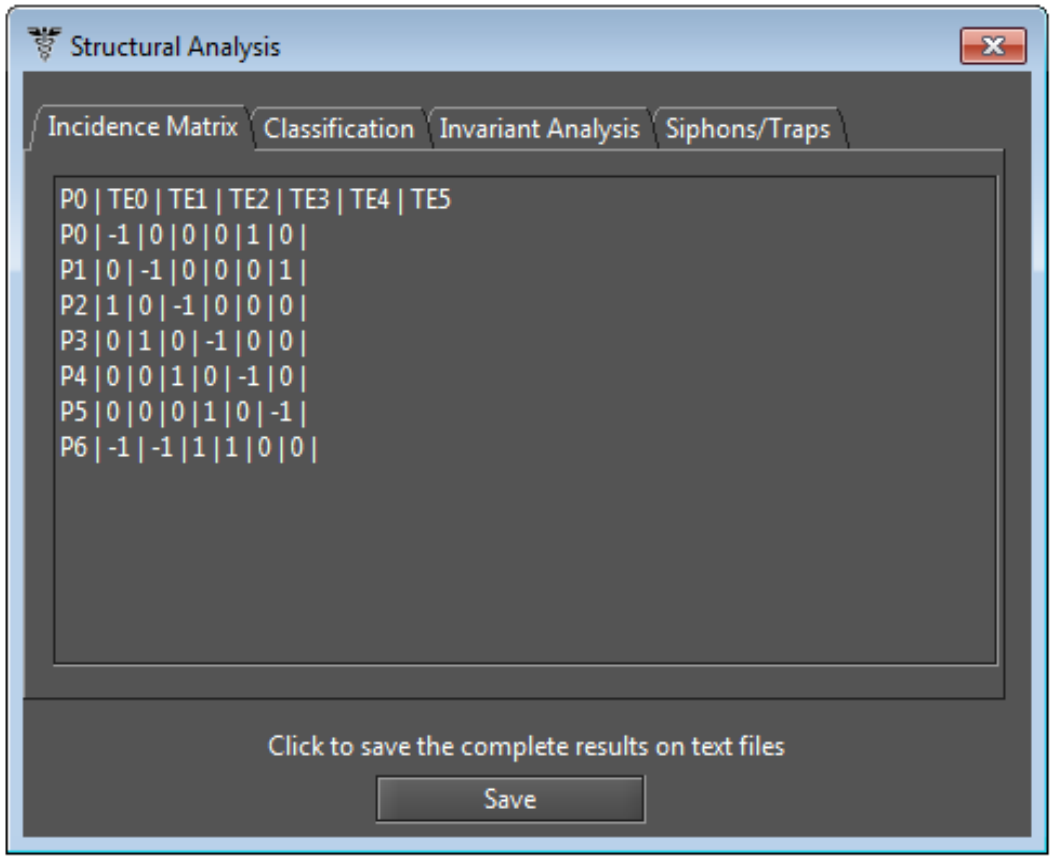
Estudo de caso – Communicating System Modeling

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



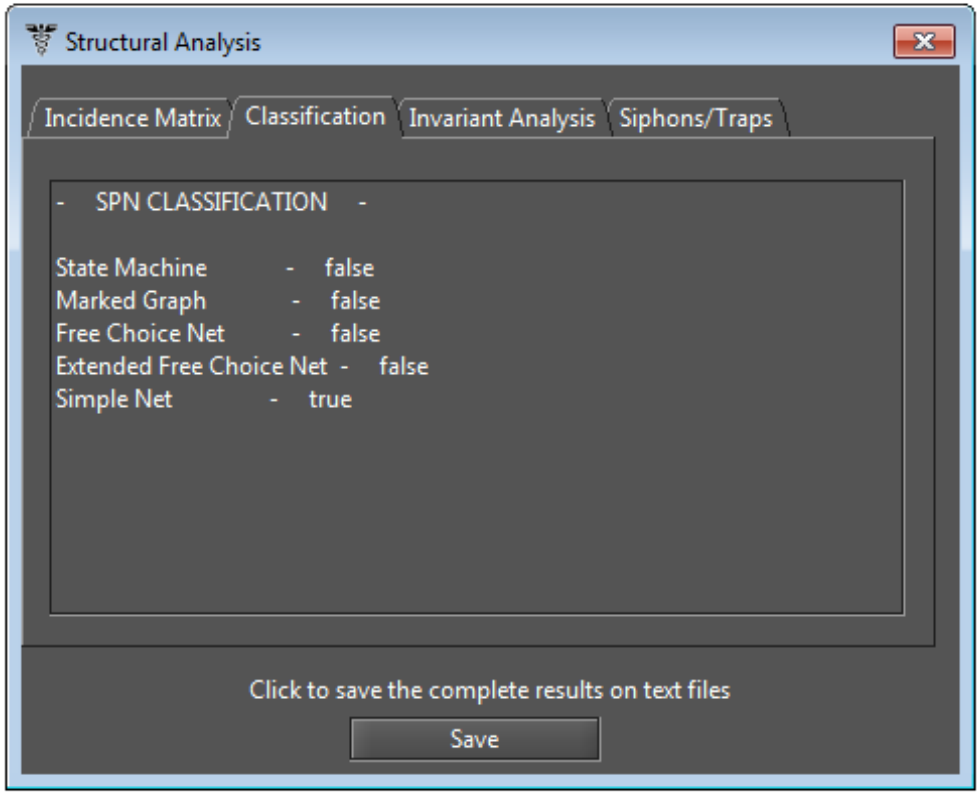
New Feature – Propriedades estruturais

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



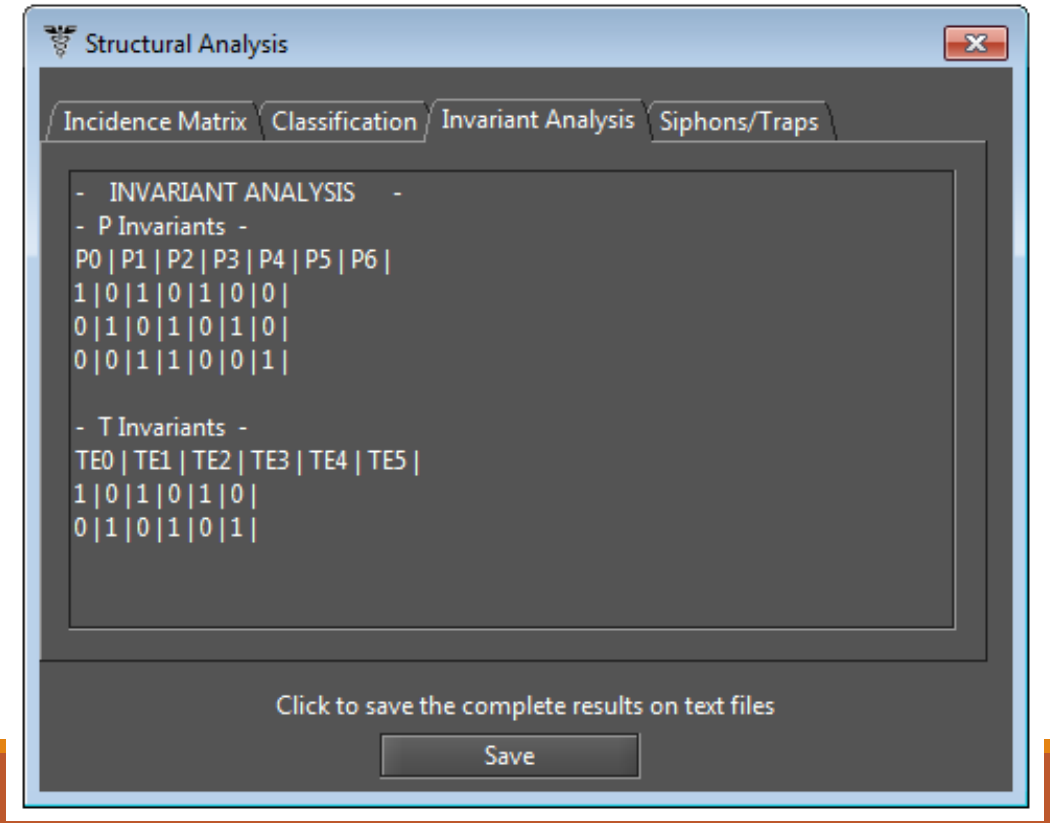
New Feature – Propriedades estruturais

- 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