INSTITUTO SUPERIOR DE CIÊNCIAS DO TRABALHO E DA EMPRESA

Departamento de Ciências e Tecnologias da Informação

EXTENSÃO DA BIBLIOTECA REPAST PARA DESENHO EM TEMPO REAL DE REDES DE PETRI EM REPRESENTAÇÃO DE SIMULAÇÕES MULTI-AGENTE

Manuel Leal e Sousa

Tese submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia Informática e de Telecomunicações

Orientador:

Professor Jorge Louçã,

Instituto Superior de Ciências do Trabalho e da Empresa

Maio, 2008

EXTENSÃO DA BIBLIOTECA REPAST PARA DESENHO EM TEMPO REAL DE REDES DE PETRI EM REPRESENTAÇÃO DE SIMULAÇÕES MULTI-AGENTE

Manuel Leal e Sousa

- Lombada –

Resumo

A utilização de plataformas de modelação e de simulação baseada em agentes tem aumentado ultimamente em comunidades de investigação de diversos domínios científicos. A ferramenta *Repast* (*Recursive Porous Agent Simulation Toolkit*) é uma das plataformas mais utilizadas. A biblioteca é de utilização livre e o seu código fonte é fornecido gratuitamente, o que facilita a sua extensão a novas funcionalidades. Algumas das características ausentes nesta ferramenta, que poderiam auxiliar o estudo de simulações, são as noções de concorrência e de paralelismo. As representações e modelações gráficas destas características, na simulação multi-agente, podem ser de grande utilidade. Este trabalho tem como objectivo colmatar esta questão através de uma extensão ao *Repast* de forma a permitir o desenho em tempo real de Redes de Petri em representação de simulações multi-agente.

Palavras-chave: Simulação multi-agente, Redes de Petri, Repast, Programação orientada a aspectos.

Abstract

The use of agent-based modelling and simulation toolkits has been increasing lately in the research communities of various scientific fields. The tool Repast (Recursive Porous Agent Simulation Toolkit) is one of the most widely used platforms. The library is a free open source toolkit, which facilitates its extension to new features. Some of the features absent in this tool, which could help the study of simulations, are the concepts of competition and parallelism. The modelling and graphical representations of these features, in multi-agent simulation, can be very useful. This thesis aims to remedy this issue by conducting an extension to Repast, to design real-time Petri Nets representing multi-agent simulations.

Keywords: Multi-agent simulation, Petri Nets, Repast, Aspect oriented programming

Agradecimentos

Em primeiro lugar gostava de expressar a minha gratidão ao Professor Jorge Louçã, orientador desta dissertação. Agradeço pela sua incondicional disponibilidade que sempre demonstrou, pela sua ajuda na elaboração deste projecto, nomeadamente as suas sugestões, conselhos, correcções, comentários e apoio técnico durante todas as etapas de desenvolvimento, enriquecendo e valorizando esta dissertação.

Queria igualmente agradecer à minha família e amigos pela compreensão e apoio evidenciado durante o período em que decorreu este projecto.

Índice

Índice de Figuras

1. INTRODUÇÃO

Esta dissertação tem como objectivo principal interligar dois domínios distintos, Redes de Petri e Simulação Multi-Agente, com o intuito de associar conceitos de cada um deles e ter como resultado final uma ferramenta mais completa de estudo de problemas simulados.

O objectivo da Simulação Multi-Agente é ir além da representação de agentes como entidades discretas e auto-contidas. Cria reproduções de uma visão de actores sociais, sendo estas reproduções permeáveis, intercaladas e que se definem mutuamente. É pretendido a modulação credível de sistemas, agentes, organizações e instituições como construções sociais recursivas.

A Rede de Petri ou Rede de Lugares e Transições constitui uma das diferentes representações matemáticas existentes para sistemas distribuídos discretos. É uma linguagem de modelação que é afigurada com grafos anotados e direccionados a estruturas de sistemas distribuídos, concorrentes, assíncronos, paralelos, determinísticos e não determinísticos.

A dissertação tem, pois, como tema de estudo a normalização da modelação por Redes de Petri como um módulo de consequências ou resultados de simulações realizadas em aplicações multi-agente, nomeadamente na plataforma Repast. Este módulo é considerado uma ferramenta auxiliar às interpretações dos desfechos dessas mesmas simulações.

Como protótipo de teste do módulo de Redes de Petri resultantes de simulações, foi implementado o problema do "Jantar de Filósofos" em simulação multi-agente Repast. Este famoso exemplo, idealizado por Edsger Dijkstra, é um enigma esclarecedor e clássico de concorrência e sincronização de sistemas multi-processo É igualmente um problema ideal para a representação em modelação por Redes de Petri.

A dissertação está organizada do seguinte modo. O primeiro Capítulo constitui a presente Introdução. No segundo Capítulo, é apresentado o estado de arte do trabalho sobre Redes de Petri e Simulação baseada em Agentes com as definições dos respectivos domínios aprofundados. No terceiro Capítulo, é apresentada uma descrição detalhada do modelo a propor para a resolução do problema em causa. O quarto capítulo consiste na validação da proposta com os respectivos testes, simulações e protótipos realizados. Finalmente, no quinto e último Capítulo, são descritas as conclusões do trabalho realizado, expondo os argumentos contra e a favor. No final da dissertação são apresentados os anexos acessórios ao documento e são enumeradas as referências bibliográficas seguidas ao longo do trabalho.

2. ESTADO DA ARTE

Neste capítulo, é efectuada uma síntese de conhecimentos sobre as Redes de Petri e simulação baseada em agentes, bem como sobre a documentação utilizada na elaboração deste trabalho de investigação. É descrito o que está a ser feito actualmente em matéria de modelação por Redes de Petri, bem como o que está a ser presentemente realizado no campo do estudo da modelação multi-agente. E, finalmente, é estudada a interacção entre estes domínios distintos.

2.1 Redes de Petri

As Redes de Petri surgiram em 1962 como resultado da tese de doutoramento do matemático alemão Carl Adam Petri. Desde então, estas redes têm sido aceites como uma ferramenta imprescindível de especificação para uma vasta variedade de sistemas, incluindo sistemas concorrentes, distribuídos, assíncronos, paralelos, determinísticos e não determinísticos. As Redes de Petri podem também ser aplicadas num extenso número de diferentes actividades, incluindo a engenharia, a química, as matemáticas, a indústria, o mundo dos negócios e até os sistemas judiciais.

2.1.1. Rede de Petri Clássica

Uma Rede de Petri ou também denominada de Rede de Lugares e Transições [Cost et al., 1999] é principalmente uma ferramenta de modelação gráfica e matemática [Lin & Lin, 2006]. É uma linguagem matemática de modelação utilizada para a representação de sistemas distribuídos discretos [Louçã, 2006]. Consiste num conjunto finito de lugares, transições, marcas (com a respectiva marcação inicial) e ligações (arcos) que ligam lugares a transições, onde cada nó da rede ou é um lugar ou uma transição. A Figura 1 mostra um exemplo simples de uma Rede de Petri.

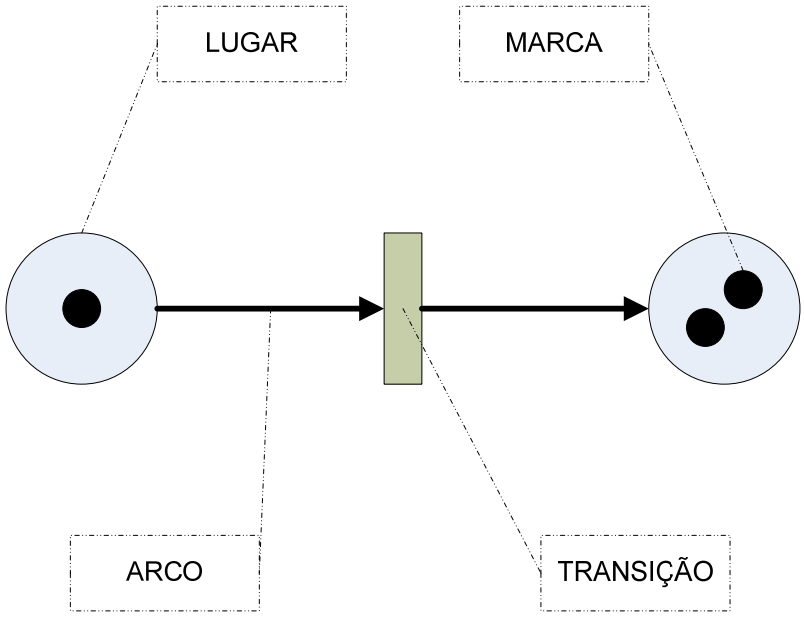


Figura 1. Uma Rede de Petri simples

Num modelo de Rede de Petri existem os seguintes constituintes:

- Lugar – O lugar é apresentado na figura como um círculo oco que representa um recurso específico ou uma determinada condição. É dito que se um lugar é um membro de função de entrada de uma transição, é pré-condição da transição, sendo pós-condição no caso de ser membro de função de saída [Brink, 1996].
- Marca – A marca é apresentada como um círculo mais pequeno e sólido que se encontra dentro de lugares e representa as condições / estado do respectivo recurso. Marcas ocupam lugares. Uma marca significa que um determinado lugar ou condição ou recurso se encontra no estado válido ou seja, activo.
- Transição – A transição é apresentada como um barra e simula um evento ou uma actividade no modelo. Este evento só poderá ocorrer quando todos lugares de entrada da respectiva transição contenham marcas suficientes para activar os designados arcos de entrada.
- Arco – Os arcos constituem as ligações directas e representam relações de entrada e saída entre um lugar e uma transição. Relacionam transições com lugares e lugares com transições. Um arco contém também um peso associado que indica o número de marcas necessárias para o activar [Brink, 1996].

A distribuição de marcas nos vários lugares representa as condições actuais (estado actual) do modelo da Rede de Petri. As transições são disparadas por regras predefinidas em conjunção com o número / quantidade de marcas dos seus lugares de entrada. Quando uma transição é disparada, isto é, quando todos seus lugares de entrada contenham marcas suficientes para a activar, as marcas irão dos lugares de entrada da transição para os de saída (fluxo de marcas). O número de marcas dentro de lugares estará sempre em constante mudança até que não existam mais transições a serem disparadas. Quando existe pelo menos uma marca em todos os lugares que estão ligados a uma transição, a transição é dita activa [Cost et al., 1999].

2.1.1.1. Definição matemática de Redes de Petri

Uma definição matemática pode completar a descrição anterior [Peterson, 1981]:

Definição matemática de Rede de Petri

Uma Rede de Petri (grafo ou estrutura) é um grafo pesado bipartido (P, T, A, w), onde:

- $P = \{p_1, p_2, ..., p_n\}$ é o conjunto finito de lugares.
- $T = \{t_1, t_2, ..., t_m\}$ é o conjunto finito de transições.
- $A \subseteq (P \times T) \cup (T \times P)$ é o conjunto de arcos de lugares para transições (p_i, t_j) e transições para lugares (t_j, p_i).
- $w: A \to \{1,2,3,...\}$ é a função de pesos associados aos arcos.

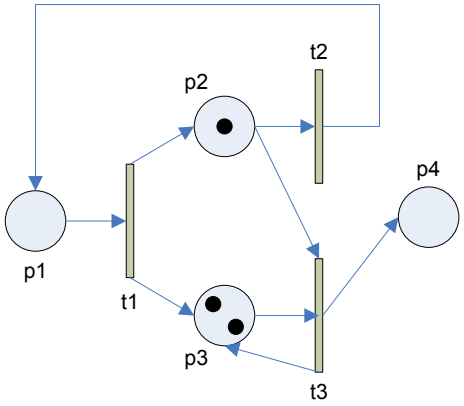
Um lugar pode pertencer a cada um dos seguintes conjuntos (ou ambos):

- Conjunto de lugares de entrada de $t_j \in T$
  - $I(t_j) = \{p_i \in P : (p_i, t_j) \in A\}$
- Conjunto de lugares de saída de $t_j \in T$
  - $O(t_j) = \{p_i \in P : (t_j, p_i) \in A\}$

Uma Rede de Petri marcada é um 5-túpulo (*P, T, A, w, x*), em que (*P, T, A, w*) é um grafo de Rede de Petri e *x* é uma marcação do conjunto de lugares *P*

- $\mathbf{x} = [x(p_1), x(p_2), \ldots, x(p_n)] \in \mathrm{N}^n$

Um exemplo de notação de uma Rede de Petri é apresentado na Figura 2:



$P = \{p_1, p_2, p_3, p_4\}$
$T = \{t_1, t_2, t_3\}$
$A = \{(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_3),(t_1,p_2),(t_1,p_3),(t_2,p_1),(t_3,p_3),(t_3,p_4)\}$
Todos os pesos são $= 1$
$x = x_0 = [0\ 1\ 2\ 0]$

Figura 2. Notação de Rede de Petri

2.1.1.2. Relação entre Redes de Petri e Sistemas Multi-Agente (SMA)

Nesta secção são descritas as relações entre Redes de Petri e os conceitos de sistema Multi-Agente (SMA). As Redes de Petri, com as suas generalizações de alto nível e hierarquias, possuem capacidades naturais para a modelação de processos concorrentes, integrados e distribuídos ou actividades num ambiente concorrente e colaborativo [Zha et al., 2003]. Estas incluem dependências temporais, concorrência, actividades (eventos) assíncronos, restrições impostas em recursos, e a própria heterogenia desses recursos. Além disso, características como a formalização, visualização, execução e avaliação, apelam para modelação, planificação e simulação de processos. Adicionalmente, as Redes de Petri são ferramentas ideais para serem incorporadas numa plataforma de Inteligência Artificial (AI). São plataformas com representação uniformizada do conhecimento, automatização no seu raciocínio e tomada de decisão. Podem facilmente ser utilizadas na modelação intensiva do conhecimento e na simulação de um ambiente concorrente.

As Redes de Petri já foram utilizadas por alguns autores na modelação de SMA. Diferentes aproximações podem ser encontradas na aplicação das redes [Fernandes & Belo, 1998]:

1. Uma primeira perspectiva visualiza uma Rede de Petri como apenas um agente. A Rede de Petri é utilizada como um diagrama que representa os vários estados internos do agente em causa.
2. Uma segunda aproximação associa um agente a cada transição ocorrida no sistema e a rede inteira representa todo o SMA. Neste caso, o comportamento de todo o sistema é obtido pela rede de ocorrências (grafo de profundidade). O grafo de profundidade ou grafo de alcance é a representação do conjunto de estados alcançáveis pela rede.
3. Numa terceira perspectiva, o comportamento de cada agente individual é descrito por uma Rede de Petri separada. Para representar todo o SMA é usado um mecanismo de combinação (junção) das várias Redes de Petri. Usualmente, para executar esta junção, as técnicas utilizadas são a sobreposição de lugares ou a sobreposição de transições.
4. Pode existir uma perspectiva semelhante à anterior, diferenciada apenas na maneira de junção / combinação das várias Redes de Petri. Nesta aproximação é utilizado a noção de Redes de Petri hierárquicas. Nestas redes, cada agente é uma Rede de Petri e as interligações entre agentes são descritas por uma Rede de Petri que representa o nível seguinte da hierarquia. As redes hierárquicas são descritas mais pormenorizadamente na secção 2.1.2.3.
5. Finalmente a última perspectiva contempla o uso de Redes de Petri Objecto para a modelação do SMA. Nesta aproximação, as marcas são consideradas como agentes. Essencialmente, Redes de Petri Objecto (OPNs) são Redes de Petri Hierárquicas que utilizam sub-Redes de Petri como marcas.

2.1.1.3. Modelação de problemas conhecidos

Nesta secção, são apresentados dois exemplos de possíveis modelações de problemas conhecidos. Estes exemplos práticos dão uma maior percepção das possibilidades existentes na utilização da simulação / modelação de Redes de Petri em problemas reais.

A Figura 3 mostra um exemplo de uma Rede de Petri que modela o conhecido problema Produtor – Consumidor. No lado esquerdo está a entidade produtora e no lado direito, a entidade consumidora. No centro, está o "armazém" de itens produzidos, permitindo o armazenamento até 16 itens.

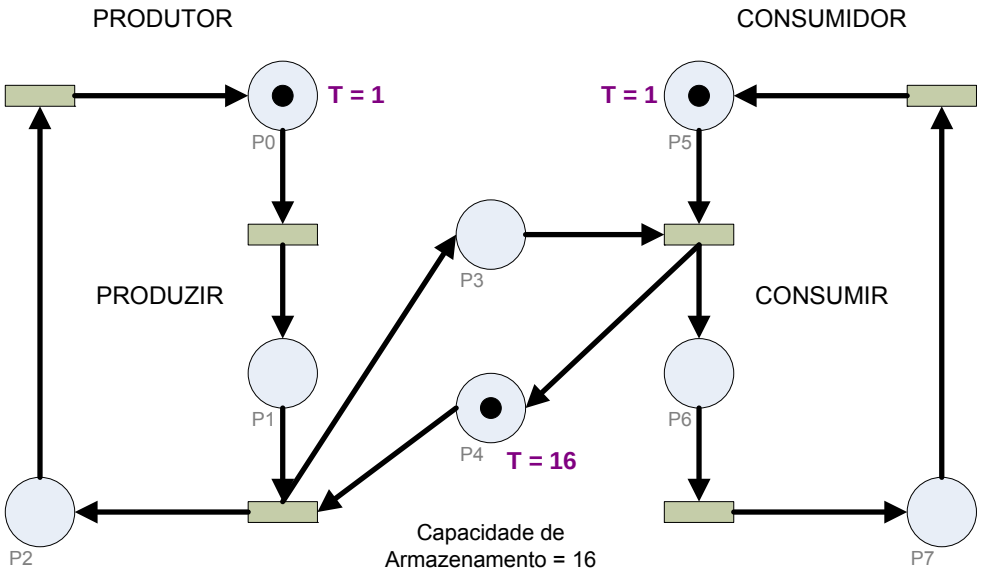


Figura 3. Exemplo da Rede de Petri do problema Produtor – Consumidor

Todos os arcos neste exemplo são activados por apenas uma marca. A marcação inicial para esta Rede é [1, 0, 0, 0, 16, 1, 0, 0]. Esta notação corresponde ao número inicial de marcas em cada lugar de P0 a P7 respectivamente. Existe uma legenda em cada lugar que contenha marcas, que indica o respectivo número de marcas lá existentes (por exemplo: T = 16).

O segundo exemplo é o famoso problema do Jantar de Filósofos [Louçã, 2006]. Na ciência informática, este problema idealizado por Edsger Dijkstra é um exemplo ilustrativo de um problema comum: concorrência. É um enigma clássico de sincronização de sistemas multi-processo.

Cinco filósofos estão sentados à volta de uma mesa circular e cada um tem um prato de esparguete à sua frente com um garfo de cada lado (Figura 4). A actividade de um filósofo consiste em comer ou pensar. Cada filósofo necessita de dois garfos para comer. Os filósofos nunca falam entre eles, o que cria a possibilidade de chegar a "becos sem saída", o que acontece quando cada um dos filósofos está a segurar no seu garfo esquerdo e à espera do direito ou vice-versa. O objectivo é evitar estes "becos sem saída" e claro, evitar, também, a fome dos filósofos.

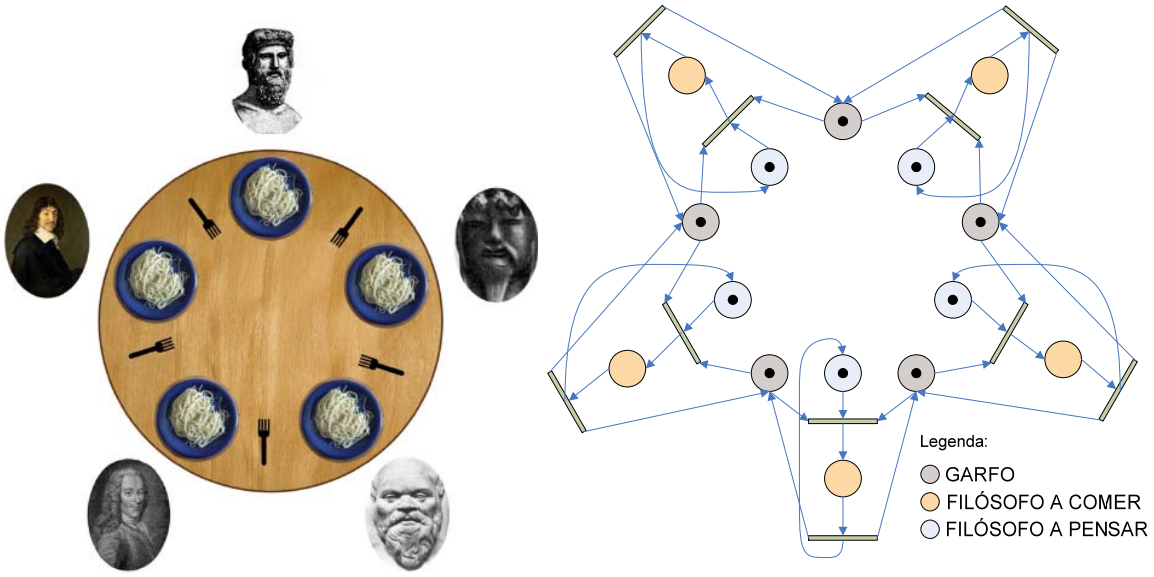


Figura 4. Modelação do Problema do Jantar de Filósofos

O Problema do Jantar de Filósofos, aqui modelado através de uma Rede de Petri, será retomado para ilustrar as seguintes extensões de RP e as propostas apresentadas nesta dissertação.

2.1.2. Extensões de Redes de Petri

Esta secção expõe de forma sucinta três extensões às Redes de Petri. Estas têm o objectivo de aumentar o poder computacional das redes tradicionais. Permitem o acrescentar de informação e uma superior estruturação [Maciel et al., 1996]. As extensões são as seguintes:

- Redes Coloridas
- Redes Temporizadas
- Redes Hierárquicas

2.1.2.1. Redes de Petri Coloridas

As redes coloridas obtiveram o seu nome pelo facto de permitirem o uso de marcas coloridas que transportem dados (valores). As marcas apresentam cores diferentes para que possam ser distinguidas umas das outras. Em contraste, nas Redes de Petri "originais", as marcas são desprovidas de cor sendo desenhadas a cor preta [Jensen, 1998].

Na rede colorida, cada marca contém um valor, sendo esse valor tipificado. A cor de uma marca é o esquema ou a especificação do seu tipo. Tipos de dados complexos e arbitrários podem

ser usados, como por exemplo, listas de vários registos constituídos por campos de tipos diferentes [Bakam et al., 2001]. Assim, as marcas são conjuntos de valores chamados multi-conjuntos. Adicionalmente, os arcos contêm o esquema que transportam e podem igualmente especificar expressões booleanas básicas. Especificamente, os arcos de saída de um lugar podem ter funções associadas que determinam quais os multi-conjuntos que são para remover, ou, caso sejam arcos de entrada de um lugar, determinam quais os multi-conjuntos que são para depositar. Assim, simples expressões booleanas denominadas de guardas são associadas a transições, para reforçar algumas restrições nos elementos multi-conjuntos [Cost et al., 1999]. Nas Redes de Petri coloridas, para uma transição estar activa, é obrigatório que, além da quantidade necessária de marcas (como nas RP originais), essas mesmas marcas satisfaçam as expressões dos respectivos arcos de entrada da transição. Na Figura 5, é apresentado um exemplo simples onde são visíveis as características agora descritas. No modelo ilustrado, a transição t_1 está habilitada visto que o lugar p_1 possui uma marca de cor "a" e o lugar p_2 contém marcas "a" e "z", existindo assim a satisfação das condições exigidas pelos arcos de entrada da transição t_1. No caso da transição t_0, as condições não são satisfeitas, como se verifica pela figura, tornando a transição não habilitada. Por fim, pela transição t_1, é depositado no lugar p_4 uma marca "v", visto ser a cor definida pelo arco de saída da t_1 [Maciel et al., 1996].

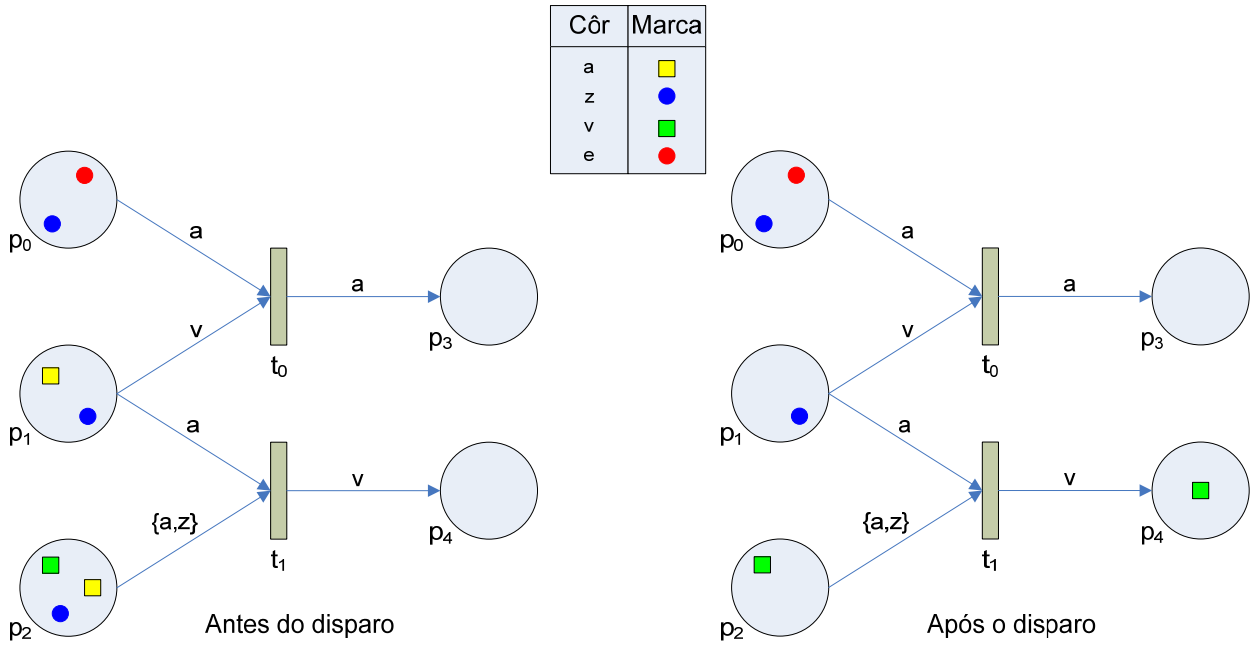


Figura 5. Rede de Petri Colorida Simples

Nos sistemas do mundo real, frequentemente são encontrados componentes similares entre si. Nas Redes de Petri clássicas, estes componentes similares são e terão que ser representados por sub-Redes de Petri disjuntas e idênticas. Isto transforma toda a Rede de Petri em algo extremamente alargado e complexo, dificultando a sua leitura. As Redes de Petri coloridas possibilitam uma representação mais compacta. As ditas sub-Redes de Petri individuais são substituídas por apenas uma sub-Rede de Petri colorida. Esta nova rede será constituída por diferentes tipos de marcas, onde cada marca terá a sua cor e corresponderá uma sub-Rede de Petri distinta na equivalente Rede de Petri original [Elkoutbi & Keller, 1998].

As Redes de Petri coloridas (RPC) são formalmente equivalentes às tradicionais Redes de Petri (RP). No entanto, a notação mais rica permite a possibilidade de modelar interacções que seriam impraticáveis nas redes tradicionais [Cost et al., 1999]. Portanto, as redes coloridas combinam elementos de programação de alto nível com o melhor das Redes de Petri clássicas [Weyns & Holvoet, 2002]. Uma definição matemática pode completar a descrição anterior [Maciel et al., 1996]:

Definição matemática de Rede de Petri colorida

Uma Rede de Petri colorida é um 9–túpulo (∑, P, T, A, w, C, G, E, I) onde:

- ∑ é o conjunto finito não vazio de tipos denominado por "conjuntos de cores". Os conjuntos de cores determinam os tipos, operações e funções que existem e podem ser usados na Rede.
- Como nas redes clássicas P é o conjunto finito de lugares, T o conjunto finito de transições e A o conjunto finito de arcos, em que e w é a função de nó (peso).
- $C{:}P \to \Sigma$ é a função de cor.
- $G{:}T \to \exp$ é a função de guarda onde *exp* é uma expressão tal que $\forall t \in T : [Type(G(t)) = Bool \wedge Type(Var(G(t))) \subseteq \Sigma]$
- $E{:}A \to \exp$ é a função associada aos arcos onde *exp* é uma expressão tal que $\forall a \in A : [Type(E(a)) = C(p(a)) \wedge Type(Var(E(a))) \subseteq \Sigma]$ onde *p(a)* é um lugar de *N(a)*.

  I é a expressão de inicialização onde $\forall p \in P : [Type(I(p)) = C(p) \wedge Var(I(p)) = \varnothing]$

Na secção 2.1.1.3, o problema do Jantar dos Filósofos foi modelado através de uma Rede de Petri clássica complexa, imensa e repetitiva. O aumento do número de filósofos tornaria a rede ainda maior e mais complexa alterando a sua estrutura (apresentando ainda mais estados

repetitivos). A sua substituição por uma Rede Colorida é bastante adequada, eliminando os processos repetitivos. Faz com que a rede seja escalável para que o aumento do número de filósofos possa ser feito simplesmente alterando a marcação inicial de filósofos e garfos [Penha et al., 2004].

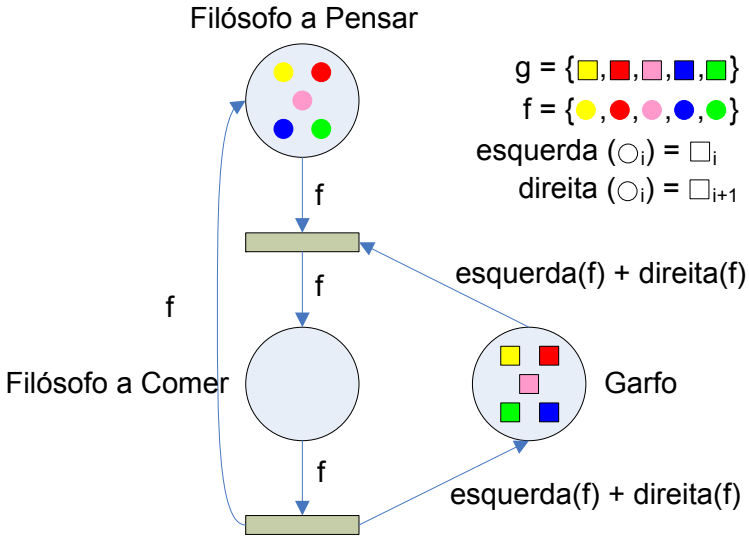


Figura 6. Modelação do Jantar de Filósofos em Rede de Petri Colorida

Nesta rede (Figura 6), cada conjunto de lugares é substituído por um único lugar contendo marcas coloridas. A fusão dos lugares resulta na fusão dos respectivos arcos. Para isto, são atribuídas funções aos arcos de forma a determinar quais as marcas que devem ser adicionados ou retirados dos lugares – funções esquerda() e direita(). Cada filósofo e cada garfo são caracterizados por cores e formas diferente. A função esquerda (filósofo $_i$) selecciona o garfo à esquerda do filósofo $_i$ [Penha et al., 2004].

2.1.2.2. Redes de Petri Temporizadas

As redes apresentadas anteriormente não incluem a noção de tempo. Este conceito foi propositadamente evitado por C.A. Petri. Isto porque, segundo ele, a associação de restrições temporais ao modelo podem impedir o disparo de certas transições. Caso isto acontecesse, seria a contradição da importante suposição (de Petri) de que uma rede representa todos os possíveis comportamentos de um sistema real. Foi concluído na altura, que, na modelação, simulação e análise de Redes de Petri, a temporização não era relevante [Marsan et al., 1995].

Cedo foi percebido que afinal o conceito de tempo era extremamente importante em aplicações onde a eficiência era relevante na modelação do problema [Marsan et al., 1995]. As primeiras ideias sobre Redes de Petri temporizadas surgiram durante a década de 1970, por P.M.Merlin e D.J.Farber [Bouyer et al., 2006].

As Redes de Petri clássicas permitem a reprodução de sistemas dinâmicos com eventos discretos. No entanto, para que seja possível modelar processos em tempo real, calcular desempenhos e examinar questões de escalonamento, é necessário consultar informação temporal que ocorre nos eventos do sistema [Marranghello, 2005]. As Redes de Petri Temporizadas (RPT) são consideradas úteis para a avaliação da performance de sistemas, particularmente em sistemas relacionados com a indústria de produção e manufactura [Barad, 1998]. A adição da noção de tempo nas Redes de Petri permite, além da modelação da lógica dos sistemas, a modelação de relações temporais [Bressan, 2002].

As RPT são extensões que adicionam a capacidade de análise temporal [Marranghello, 2005]. Nas várias abordagens sobre as redes RPT, no formato em que os tempos são especificados, existem as seguintes propostas [Maciel et al., 1996]:

1. Transições – Associação de tempo a transições. A especificação do tempo de disparo da transição. Esta indicação de tempo refere-se ao tempo dispendido pelo processamento da acção ou evento modelado pela transição [Marranghello, 2005]. Em certas reproduções gráficas de redes RPT, as transições imediatas, sem tempo associado, são representadas por barras finas. Caso sejam transições temporizadas são representadas por barras grossas [Haas, 2004].
2. Lugares – Associação de tempo a lugares. Nos lugares de saída temporizados, as marcas aí armazenadas só estarão disponíveis para disparar transições após o acabar do tempo associado ao lugar [Maciel et al., 1996]. Por outras palavras, o tempo associado a um lugar corresponde ao período em que a marca deve permanecer naquele lugar até ser novamente disparada [Marranghello, 2005].
3. Marcas – Associação de tempo a marcas. Cada marca possui individualmente uma informação temporal indicando a disponibilidade de disparo de uma transição [Maciel et al., 1996]. Neste caso as marcas podem carregar consigo informação adicional correspondente a temporizações.

4. Arcos – Associação de tempo a arcos. Nesta temporização, em cada arco é especificado um respectivo tempo de disparo [Marranghello, 2005].

Dado um determinado modelo onde as temporizações estão associadas a transições, é possível criar um outro modelo equivalente, onde as temporizações já estarão associadas a lugares e vice-versa. Isto é ilustrado na Figura 7 [Barros, 1996].

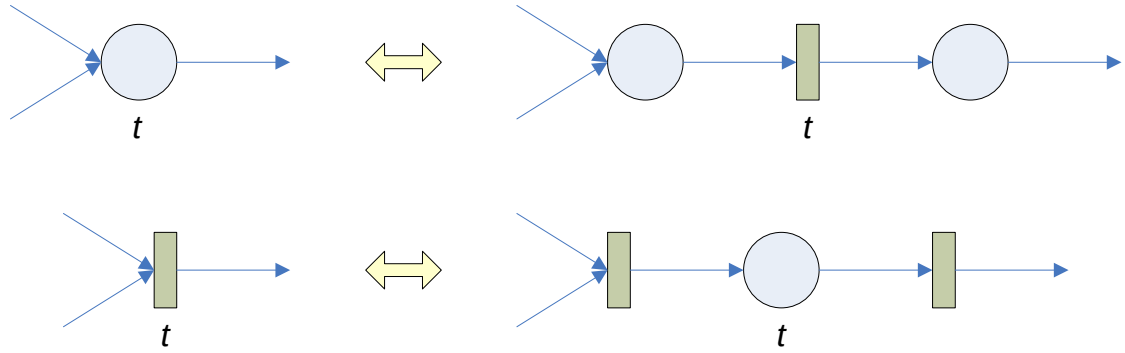


Figura 7. Igualdade entre Redes de Petri Temporizadas

Nas redes RPT existem duas tipificações em relação ao modo de utilização da informação temporal:

- Determinísticas – Determinação de temporizações absolutas ao processamento de eventos correspondentes [Marranghello, 2005]. Neste tipo de redes são especificados apenas intervalos de tempo fixos. São ideais para a modelação de sistemas onde as relações temporais existentes são determinísticas ou constantes [Penha et al., 2004]. A sua definição matemática é a seguinte [Marranghello, 2005]:

Definição Matemática de Rede de Petri temporizada determinística
Uma Rede de Petri determinística é um 6–túpulo (P, T, A, w, x, I), onde os primeiros 5 parâmetros correspondem a uma Rede de Petri clássica e o sexto parâmetro I é o conjunto de tempos associados a objectos da rede (geralmente transições). As temporizações correspondem a intervalos de tempo fechados estáticos.

- Estocásticas – Consideração de incertezas na execução de acções no sistema, associando aos eventos do modelo funções de probabilidade no cálculo das temporizações associadas [Marranghello, 2005]. Desta forma, é uma rede probabilística e aleatória, descrevendo um processo estocástico. A sua definição matemática é a seguinte [Penha et al., 2004]:

Definição Matemática de Rede de Petri temporizada estocástica

Uma Rede de Petri temporizada estocástica é um 6-túpulo (*P, T, A, w, x, D*) onde também os primeiros 5 parâmetros correspondem a uma Rede de Petri clássica e o último parâmetro D é o conjunto de tempos associados às transições (ou outros objectos da rede) que seguem uma distribuição exponencial. Estas temporizações podem também depender da marcação existente.

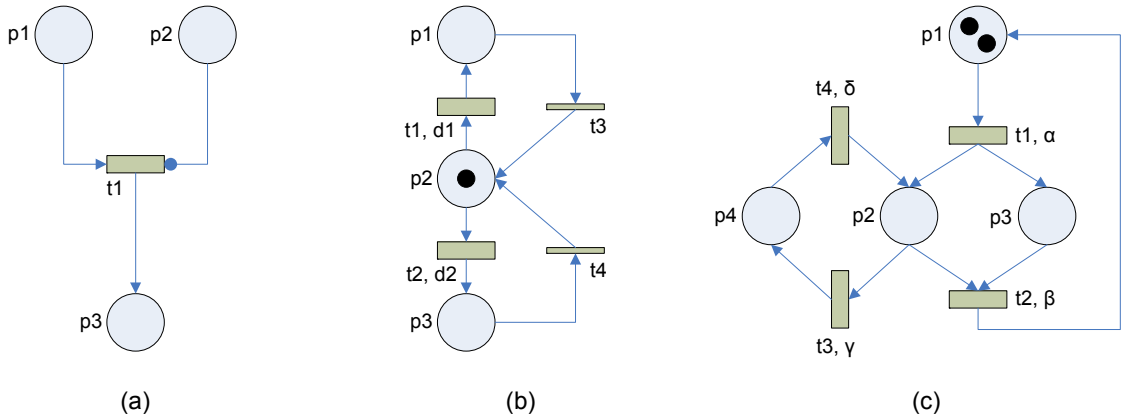


Figura 8. Várias Redes de Petri temporizadas

Na Figura 8, são apresentados três tipos de Redes de Petri. O modelo (a) com a introdução de mais uma extensão às Redes de Petri, o arco inibidor, onde é criada a possibilidade do teste zero de uma variável. A (b) que apresenta uma rede determinística com os seus intervalos de tempo estáticos e a rede (c) que especifica uma rede estocástica com temporizações aleatórias.

2.1.2.3. Redes de Petri Hierárquicas

A modelação de sistemas que apresentem alguma complexidade apenas é credível quando existem mecanismos de estruturação dessa mesma modelação. A noção de hierarquias, uma extensão às Redes de Petri, é um mecanismo usado para a reestruturação da modelação. Esta modelação hierárquica de sistemas revela os seguintes benefícios [Fernandes, 1994]:

- A possibilidade de inspecção de processos (modelos) a vários níveis de pormenor.
- O refinamento de um nó da rede (lugar ou transição) possibilitando visualizações separadas de apenas partes do sistema.
- A capacidade e facilidade de reutilização de partes da rede (do modelo).

- Os variadíssimos testes realizados numa Rede de Petri, podem agora ser efectuados hierarquicamente e isolados.
- Atenuação da problemática complexidade de uma Rede de Petri.

Este tipo de modelação diminui drasticamente o tamanho da especificação (da rede), aumentando a reutilização, reduzindo assim a ocorrência de erros na rede e melhorando a sua legibilidade [Fernandes et al., 1997].

Existem as seguintes possibilidades para a implementação da hierarquia na modelação das redes: utilização de macro nós (lugares ou transições) e a utilização de macro marcas. As macro transições e os macro lugares representam sub-redes hierárquicas, encapsulando macro eventos e macro actividades respectivamente. As macro marcas especificam hierarquias de sub-marcas, encapsulando estruturas complexas do sistema em macro marcas [Machado et al., 1997 e 1998]. Uma sub-rede completa pode ser substituída por um único lugar ou uma única transição. O extremo da abstracção e hierarquização é a substituição de toda uma Rede de Petri por apenas um nó [Fernandes, 1994]. A interface entre um macro lugar (ou lugar de substituição) e a correspondente sub-rede é realizada pelas denominadas transições porta. No caso das macro transições (ou de substituição), a interface é feito pelos lugares porta [Marranghello, 2005].

A Figura 9 exibe a utilização de macro lugares nas redes hierárquicas. A parte (a) da figura apresenta uma Rede de Petri, onde existe um lugar de substituição, ou macro lugar denominado por P. A parte (b) da figura mostra a sub-rede hierárquica correspondente. A parte (c) apresenta o resultado da substituição da Rede de Petri inferior no macro lugar da Rede de Petri principal. Nesta secção (c) é realizada a recuperação do modelo completo, desfazendo a hierarquia, efectuando a denominada planarização da rede. As transições porta são desenhadas a tons mais escuros. Na parte (b) da figura, estas transições porta são cópias das originais de entrada e de saída do macro lugar no modelo hierarquicamente superior. São utilizadas para situar e reflectir a Rede de Petri superior na sub-rede inferior [Marranghello, 2005].

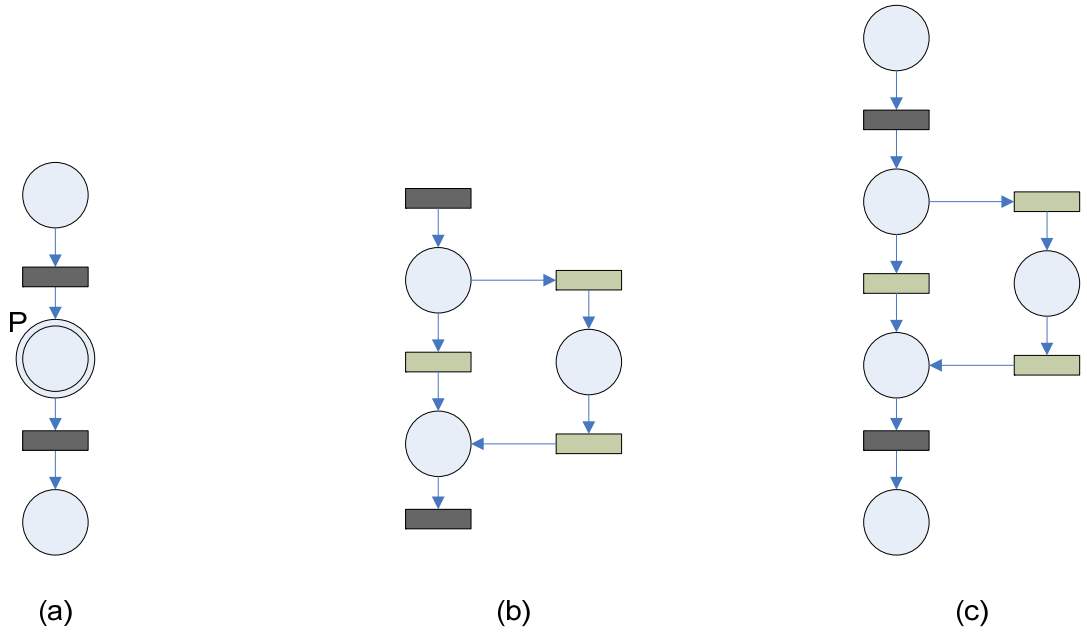


Figura 9. Utilização de Macro lugares em Rede de Petri Hierárquica

Novamente é utilizado o problema do Jantar dos Filósofos, por se tratar de um modelo complexo e repetitivo, sendo que o aumento de número de filósofos aumenta a complexidade do problema. É um excelente exemplo de utilização das redes hierárquicas, onde cada filósofo é representado no modelo por uma macro transição (Figura 10) [Barros, 1996].

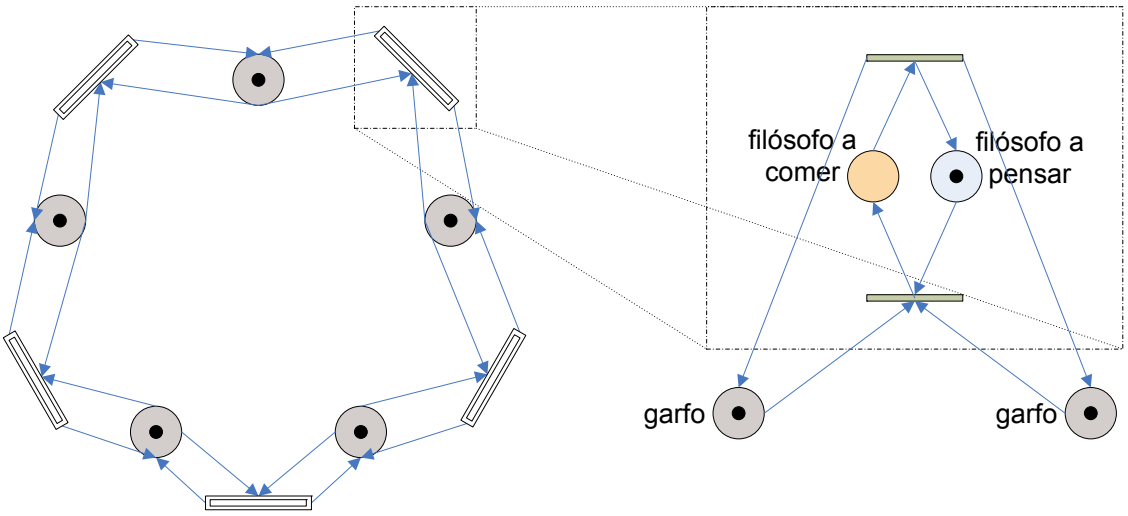


Figura 10. Modelação do Jantar de Filósofos em Rede de Petri Hierárquica

As redes hierárquicas permitem e facilitam o desenho de sistemas mais complexos, encapsulando e abstraindo partes da rede principal em redes inferiores. Estas sub-redes comportam-se como lugares ou como transições e terão de ser vivas [Barros, 1996]. Por definição, uma Rede de Petri viva é aquela onde existe sempre uma transição que possa acabar por disparar a partir de qualquer estado alcançável da Rede de Petri (assunto discutido ao pormenor na secção seguinte) [Miczulski, 2001].

2.1.3. Análise das Redes de Petri

As secções anteriores descreveram a modelação ou capacidade de esquematização de um determinado problema por Redes de Petri. No entanto, a modelação não é a única potencialidade das redes. Existe, ainda, uma série de metodologias que possibilitam técnicas de análise ao sistema (rede) em causa, verificando um vasto número de propriedades desse mesmo sistema modelado. Uma dos métodos de análise é a denominada árvore de cobertura (e grafo de cobertura). A figura 11 ilustra este processo de análise [Lima, 2001] [Maciel et al., 1996].

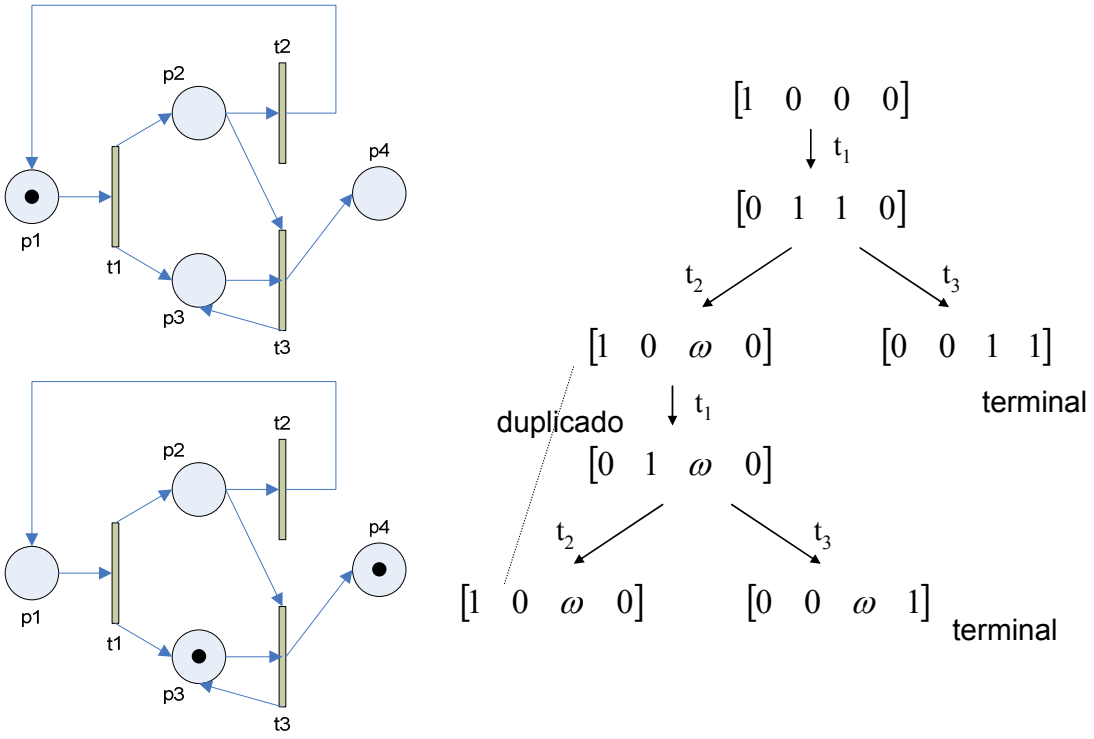


Figura 11. Árvore de cobertura

Dada uma determinada marcação inicial para uma Rede de Petri, é possível a obtenção de outras distintas marcações disparando transições potencialmente habilitadas. Como se pode verificar na Figura 11, este conjunto infinito de marcações pode ser representado pela construção de uma árvore onde os arcos são as transições disparadas e os nós as exequíveis marcações. Para efectuar a figuração finita das infinitas marcações foi utilizado o símbolo ω que pode assumir um valor infinito. Na Figura 12 é apresentado o grafo de cobertura, que ilustra o mesmo exemplo da árvore de cobertura. É uma representação alternativa à árvore de cobertura [Lima, 2001] [Maciel et al., 1996].

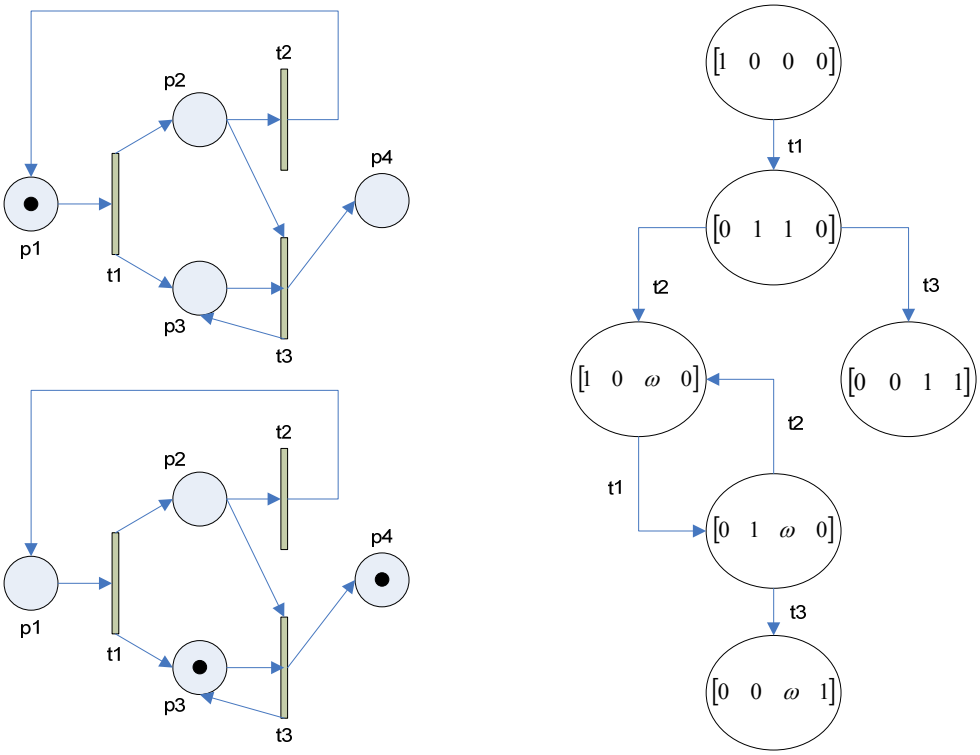


Figura 12. Grafo de cobertura

A árvore e/ou grafo de cobertura possibilita a análise das seguintes propriedades de uma Rede de Petri [Lima, 2001] [Maciel et al., 1996]:

1. Limitação – Uma rede é dita limitada caso o número de marcas em cada lugar não exceda um determinado limite k (k-limitada) em qualquer marcação atingível. Em redes limitadas, as árvores e os grafos de cobertura passam a ser denominadas de árvores e grafos de alcance devido ao facto de conterem todas as marcações acessíveis e possíveis na rede.

2. Segurança – Esta propriedade é uma "especialização" da propriedade limitação, uma vez que uma rede segura é aquela em que todos os lugares só podem ter uma ou nenhuma marca (k=1). Um lugar 1-limitado é chamado um lugar seguro. Esta análise é essencial na especificação de sistemas digitais (Exemplo: sistemas binários, *flip-flops*).
3. Vivacidade – Uma transição é viva se é potencialmente habilitada para qualquer marcação de uma Rede de Petri. A rede é viva se existir a possibilidade de disparar qualquer transição através do disparo de alguma sequência de transições, isto para qualquer marcação possível. Pelo contrário, uma transição é intitulada morta caso nunca esteja habilitada qualquer que seja a marcação da Rede. Esta propriedade é extremamente importante no estudo de possíveis bloqueios ou impasses em sistemas reais. A vivacidade permite o estudo de possíveis *deadlocks* quando existe partilha de recursos entre processos. Numa Rede Viva nunca poderá ocorrer um *deadlock*. A Figura 13 ilustra dois exemplos de vivacidade nas Redes de Petri.

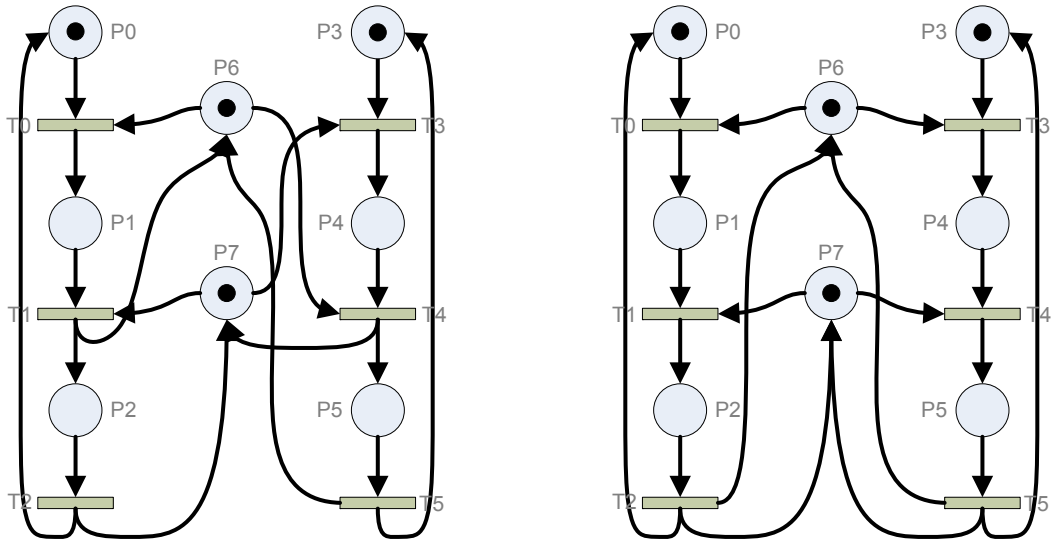


Figura 13. À esquerda – Rede com *deadlock*, à direita – Rede viva

Ambas as redes representam dois processos que partilham os recursos caracterizados pelos lugares P6 e P7. Na rede à esquerda, caso os dois processos necessitem dos recursos simultaneamente, o sistema pode bloquear. Isto é, caso seja disparada a transição T0 e, de seguida, disparada a transição T3, deixam de existir marcas suficientes para disparar qualquer outra transição. Na rede à direita, não há a possibilidade de ocorrer *deadlocks*, dado que não existe uma marcação que impossibilite o disparo de transições.

4. Conservação – A conservação é uma propriedade que assinala se o número total de marcas na Rede se mantém constante com o disparo de transições. Isto é, indica se o disparo de uma transição não altera (conserva) o número de marcas na rede. Permite, por exemplo, num determinado sistema, verificar se os recursos existentes não são removidos ao longo do tempo.
5. Alcance das marcações – A análise desta propriedade consiste em verificar se uma determinada marcação é atingida, através de um conjunto finito de disparos de transições, a partir de uma outra determinada marcação inicial. É essencial no estudo da dinâmica de um sistema. Por exemplo: saber se uma determinada marcação, que origina um impasse no processo em causa, é atingível.

Estudados os conceitos fundamentais das Redes de Petri, a correspondente análise e as respectivas extensões, é apresentada uma breve conclusão na secção seguinte.

2.1.4. Conclusão

Toda esta secção foi baseada em duas entidades distintas, Sistema e Modelo:

- Sistema – Objecto de estudo. É a colecção ou conjunto de itens / objectos relacionados de alguma forma entre si.
- Modelo – Descrição do referido sistema por meio de representação gráfica assente nos princípios que governam o sistema. Por outras palavras, é uma abstracção do sistema em causa. Esta é a definição de modelo discreto. Existem modelos analíticos formados por uma série de equações ou relações matemáticas que tentam prever o comportamento do sistema.

As simulações de modelos discretos permitem inferências sobre os sistemas modelados sem a necessidade de construí-los, quando estes são ainda apenas propostas. O exemplo estudado foi o modelo discreto Redes de Petri.

Após o estudo e análise cuidada da modelação por Redes de Petri e respectivas extensões, ficou concluído que as Redes de Petri são uma excelente ferramenta na modelação e na análise de processos. É possível identificar várias vantagens na sua utilização:

- Previne a criação de ambiguidades, incertezas ou contradições no modelo especificado, obrigando à geração de definições precisas.
- É perfeitamente possível a identificação e o estabelecimento de padrões na rede.

- Prova ser a base ideal para a importante discussão de processos.
- Todo este formalismo gera a possibilidade da utilização de estudos e técnicas analíticas sobre os modelos.

Com toda a certeza que, este tipo de modelo já foi totalmente analisado e perfeitamente testado ao pormenor, visto que as Redes de Petri foram idealizadas e criadas em 1962.

Facilmente são estabelecidas analogias entre as linguagens de programação de alto nível e as Redes de Petri. A concepção e projecção de extensões às Redes de Petri foi algo que veio tentar aumentar esta aproximação entre a programação de alto nível e a modelação das redes.

As Redes de Petri clássicas cumprem a maioria dos requisitos na especificação de sistemas. No entanto, apresentam algumas limitações, necessitando de extensões ao modelo consoante a complexidade do processo. As extensões criadas vieram aumentar o poder de modelação nas seguintes características:

1. Informação ou valor – Redes de Petri Coloridas.
2. Tempo – Redes de Petri temporizadas (determinísticas e estocásticas).
3. Hierarquia ou sub-modelação – Redes de Petri hierárquicas.

Apesar das extensões às redes clássicas aumentarem verdadeiramente o poder de modelação, simultaneamente poderão estar a reduzir o poder de decisão destas.

São indicados exemplos de algumas áreas nas quais tipicamente se encontram e se aplicam Redes de Petri:

- Sistemas de produção
- Circuitos integrados
- Automatização de trabalho de escritório
- Avaliação de desempenhos
- Protocolos de comunicação
- Simulações sociais e da sociedade
- Sistemas distribuídos
- Base de dados
- Automatização de manufactura

A modificação e constante transformação de um modelo representado por uma Rede de Petri é de simples execução. Facilmente é adicionado um evento, tarefa ou processo ao modelo em causa. Isto é extremamente útil, uma vez que qualquer sistema do mundo real nunca é

estanque. O lado esquerdo da Figura 14 retrata esta situação de uma forma engraçada e enganadora.

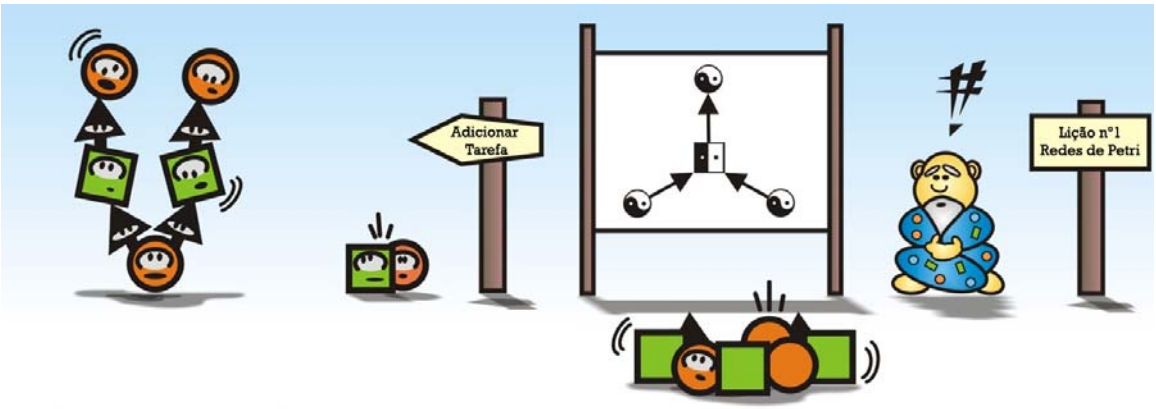


Figura 14. Caricaturas engraçadas sobre Redes de Petri [Hoheisel, 2007]

Esta modelação representada nas Redes de Petri é totalmente padrão (*standard*), independente do sistema que está a especificar e desprovida de qualquer idioma ou cultura. É possível observar uma caricatura no lado direita da Figura 14 a reflectir sobre esta situação.

A modelação de Redes de Petri é ideal na descrição de sistemas onde existam processos concorrentes e / ou paralelos. É também uma especificação que potencia estudos estatísticos sobre a informação empírica do modelo. As Redes de Petri são de extrema utilidade na modelação de sistemas multi-agente (SMA) e na reprodução gráfica de, por exemplo, configurações iniciais do sistema ou resultados da simulação do sistema. Funcionam como ferramenta de desenho do sistema e ferramenta de implementação e programação de processos.

A secção que se segue fala do conceito Simulação Baseada em Agentes. São descritas possíveis utilizações de Redes de Petri no contexto da Simulação, algo que já foi introduzido neste mesmo capítulo, na secção “Relação entre Redes de Petri e Sistemas Multi-Agente (SMA)”.

2.2. Simulação Baseada em Agentes

Nesta secção é apresentada uma breve introdução ao conceito de Simulação baseada em agentes.

2.2.1. Definição

Uma definição da palavra simulação, obtida em dicionário da Porto Editora, diz que é a "Representação de um sistema ou de um processo por um modelo com que se trabalha, como se tratasse desse sistema ou processo, para investigar os seus efeitos". Esta descrição dá com uma vaga ideia da intenção da realização de uma simulação. A enumeração dos possíveis propósitos (prognóstico, prova, descoberta, treino, entretenimento, educação e performance) da simulação ajuda a completar sua explicação [Axelrod, 2003]:

1. Prognóstico – A simulação permite, por mecanismos estudados, processar parâmetros de entrada (contextos diferenciados e complexos) e gerar resultados ou consequências.
2. Prova – Simulações servem para efectuar comprovativos de teorias ou hipóteses propostas.
3. Descoberta – No contexto da metodologia científica, a simulação assenta em três princípios: prognóstico, prova (tópicos anteriores) e descoberta. A característica "descoberta" é tão relevante quanto as outras duas. As simulações no contexto das ciências sociais são um exemplo da importância deste propósito. Neste âmbito das ciências sociais, os modelos de simulação raramente conseguem atingir uma prova final precisa ou validar um prognóstico, pelo facto das ciências sociais não serem exactas. Nem sempre é possível, por exemplo, prever com exactidão a movimentação de um determinado grupo de pessoas. No entanto, na procura da prova ou na tentativa de validação de um prognóstico, são realizadas importantes descobertas de princípios e relações durante as simulações efectuadas.
4. Treino – Existem inúmeros sistemas de simulação desenhados com o objectivo de treinar pessoas nas mais variadíssimas áreas. Sistemas que tentam representar, na máxima precisão, os processos e ambientes reais. Existem vários exemplos: Simuladores de voo, Simulacros de situações, entre outros como o que está representado na Figura 15, que data de 1915.

Figura 15. Simulador mecânico de cavalo feito em madeira (ano 1915) [Hertel, 2002]

5. Entretenimento – No seguimento do ponto anterior, esse referido treino pode tornar-se divertimento. Numa simulação é possível a aceder ambientes e mundos imaginários que reflectem realidades inatingíveis. Exemplo: *Paint ball*, onde existe a simulação de entretenimento de um ambiente de guerra.
6. Educação – Novamente, como seguimento dos pontos anteriores, as referidas características "treino" e "entretenimento" podem constituir "educação". Neste tópico, a simulação actua como potenciador de instrução e permite a aquisição de conhecimentos e aptidões por parte dos utilizadores.
7. Performance – Na óptica do domínio da inteligência artificial, a realização da simulação de tarefas (tipicamente executadas por humanos) por parte de entidades automatizadas com inteligência, tem a vantagem do aumento drástico da performance. A simulação tenta imitar a percepção, a decisão e a interacção social de um humano.

No contexto computacional, tendo como base as descrições e características enumeradas até agora, nasceu um novo conceito: Simulação baseada em agentes. A simulação multi-agente ou baseada em agentes é uma tecnologia que surgiu na ciência computacional nos últimos 20 anos. Uma possível definição de agente é: um programa computacional autónomo (processo computacional) com a capacidade independente de executar acções próprias em ambientes tipicamente imprevisíveis e dinâmicos [Wooldridge & Jennings, 1998].

A simulação baseada em agentes é uma tecnologia especialmente utilizada na resolução de problemas, sendo uma área de estudo no domínio da inteligência artificial distribuída. Tem

como principais actores os agentes, que frequentemente cooperam entre si com o objectivo de resolver problemas, ajudando-se mutuamente e trocando informações. Esta área tenta estudar estes mesmos actores, designadamente estudando o seu comportamento, as suas interacções e como cooperam ou competem para a resolução de um determinado problema. A simulação multi-agente é o suporte do desenho, especificação, implementação e investigação deste tipo de aplicações [Miranda & Perkusich, 1999].

A simulação baseada em agentes, num determinado contexto de um problema, poderá ser vista como a intersecção de três áreas científicas distintas apresentadas na Figura 16 [Davidsson, 2002].

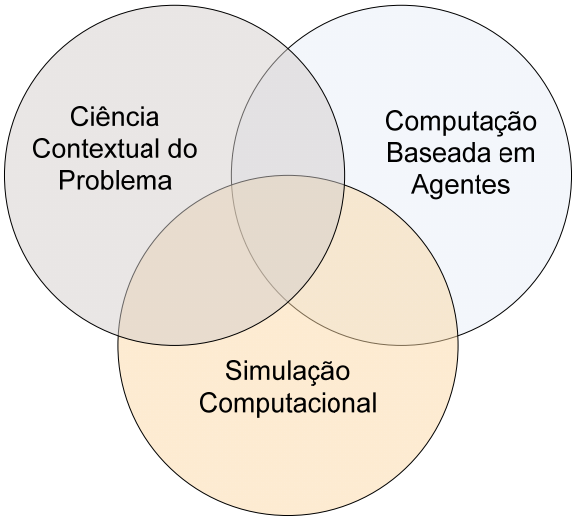


Figura 16. Áreas constituintes da Simulação Baseada em Agentes

Estes três âmbitos são caracterizados da seguinte forma [Davidsson, 2002] [Wooldridge & Jennings, 1998]:

1. Computação Baseada em Agentes – Esta área de estudo, inserida na ciência computacional, analisa o paradigma do desenho, programação e modelação de sistemas baseados em agentes. Isto é, estuda modelos e arquitecturas de agentes. Constrói e implementa agentes tendo em conta simultaneamente os processos internos e a interacção com o ambiente exterior ao agente. Compreende, também, a investigação da interacção existente entre diferentes agentes em sistemas multi-agente.
2. Ciência Contextual do Problema – Este sector engloba as diferentes ciências sobre as quais incide o problema em causa. Podem fazer referência a qualquer contexto real onde irá ser realizado o estudo da dificuldade a ultrapassar. Existem variadíssimas áreas de aplicação,

como por exemplo: processos industriais, ciências económicas, estudo de aparelhos espaciais, entre outros. Esta dissertação realça a área das ciências sociais. Estas são o conjunto numeroso de diferentes ciências que realizam o estudo da interacção entre entidades sociais da realidade (Exemplos: controlo e gestão social, psicologia social, biologia).

3. Simulação Computacional – Este âmbito compreende as diferentes técnicas existentes utilizadas na realização da simulação computacional. Exemplos: a simulação orientada a objectos, simulação matemática baseada em equações e a simulação assente em eventos discretos. No âmbito computacional, o fenómeno de simulação tem como objectivos depurar modelos e processos existentes Tenta prever futuros comportamentos de sistemas e efectua experiências que não poderiam ser realizadas no mundo real.

A área das ciências sociais é um dos âmbitos onde a simulação multi-agente é extremamente útil. Cientistas sociais têm tentado utilizar processos computacionais para representar e analisar as suas teorias, tentando clarificá-las. É-lhes possível simular processos sociais e efectuar experiências que seriam impraticáveis em ambientes reais ou que seriam difíceis de observar directamente [Davidsson, 2002].

Em qualquer simulação, toda a parametrização terá que ser exacta e não ambígua para que se obtenham modelos credíveis representativos do problema social em causa [Davidsson, 2002].

A investigação realizada em 1971 por Schelling, que propôs modelos dinâmicos relacionados com a segregação racial, é um bom exemplo de uma simulação social baseada em agentes [Schelling, 1971]. Nestes modelos, cada agente correspondeu a uma morador de um determinado bairro ou zona residencial. Existiu uma tentativa de simulação e previsão de qual a migração da localização da habitação destes residentes. Esta simulação teve em conta certos parâmetros do ambiente que rodeava os habitantes. Por exemplo, a cor predominante das casas a circundar cada agente (habitante), entre outros. Na Figura 17, são apresentadas representações gráficas do ambiente da simulação e dos respectivos agentes. É possível verificar a evolução da migração de residentes na simulação efectuada [Gilbert, 2004].

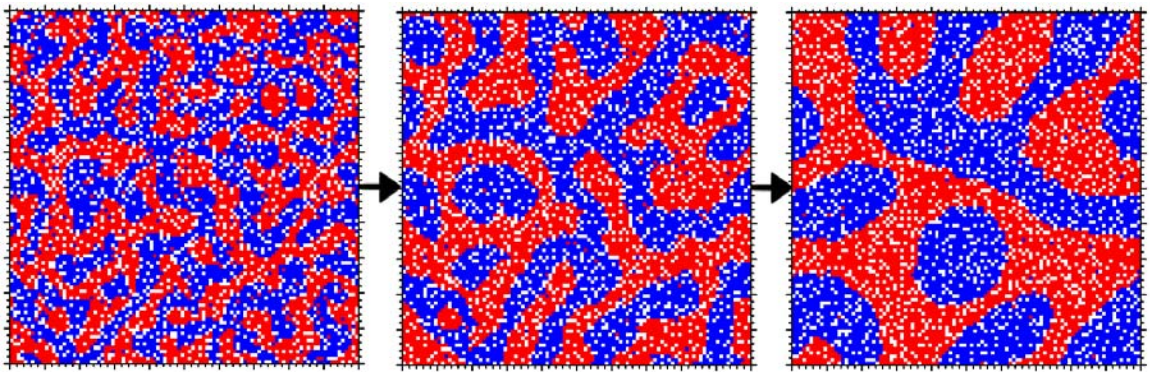


Figura 17. Modelo Gráfico de Simulação Social realizado por Schelling

Schelling propôs e quis identificar estas simulações ao conhecido problema social: a segregação racial em zonas residenciais que ocorre em determinadas cidades [Gilbert, 2004].

Com o objectivo de facilitar a elaboração de simulações baseadas em agentes, como o exemplo de Schelling, foram criados ambientes de apoio à implementação de aplicações computacionais multi-agentes. Estas bibliotecas são descritas mais pormenorizadamente na secção seguinte.

2.2.2. Ferramentas e Aplicações

Nestes últimos anos, têm sido desenvolvidas ferramentas e bibliotecas aplicacionais de simulação de sistemas multi-agente. A intenção destes instrumentos é auxiliar os investigadores no estudo de problemas e modelos baseados em agentes. São bibliotecas de componentes reutilizáveis na construção, análise e controle de modelos de simulação. Aqui, nesta secção, são referidos três bons exemplos de ferramentas aplicacionais existentes:

- *Swarm*
- *Repast*
- *Mason*

2.2.2.1. *Swarm*

Swarm é um pacote aplicacional com o intuito de ser útil a cientistas no estudo de modelos baseados em agentes. É um conjunto de bibliotecas desenvolvido no Instituto de Santa Fe (E.U.A.). Permite simulações de modelos baseados em agentes escritos na linguagem

Objective-C ou Java. É de utilização livre e o seu código fonte é fornecido gratuitamente. A tradução de *Swarm* para português é a seguinte: Comportamento manifestado colectivamente [Minar et al., 1996].

É uma plataforma de simulação multi-agente, onde a principal unidade é o conjunto formado por agentes a executar acções estipuladas, conjunto que é denominado por *swarm*. Existe a noção de herança e composição hierárquica na entidade *swarm* que potencia a reutilização de código nesta biblioteca orientada a objectos. Esta ferramenta providencia componentes para a modelação, controlo, análise e representação gráfica de problemas multi-agente [Minar et al., 1996].

A unidade básica na simulação *Swarm* é a entidade agente. Este interage via eventos discretos, sendo qualquer tipo de actor no sistema (entidade), afectando outros agentes e a si próprio. A simulação é o grupo de agentes interactivos. A Figura 18 apresenta um exemplo de uma simulação de um ecossistema. Os agentes podem representar coelhos ou coiotes [Minar et al., 1996].

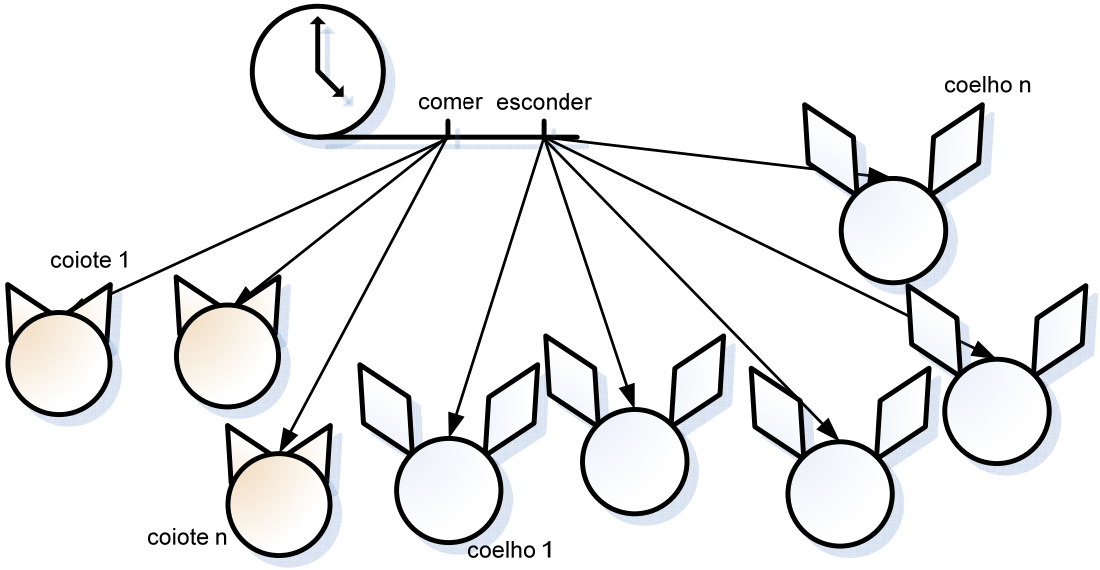


Figura 18. Coelhos e coiotes no *Swarm* [Minar et al., 1996]

No *Swarm*, acções individuais ocorrem sequencial e sucessivamente. São controladas no tempo por uma estrutura ou motor que especifica a hora e a ordem pela qual as ditas acções se executam. Como exemplo, na mesma simulação referida anteriormente, existem as seguintes acções: coiotes a comer coelhos e coelhos a esconderem-se de coiotes (motor de agendamento representado como um relógio na Figura 18) [Minar et al., 1996].

2.2.2.2. *Repast*

Repast é uma biblioteca aplicacional. A sigla *Repast* denota "*The Recursive Porous Agent Simulation Toolkit*". Esta ferramenta é, também, de utilização livre e o seu código fonte é fornecido gratuitamente. Foi desenvolvida por Sallach e outros na Universidade de Chicago [Altaweel et al., 2002].

Repast apresenta muitos conceitos da biblioteca *Swarm*. O seu principal objectivo é tentar suportar o desenvolvimento de modelos credíveis, flexíveis e recursivos de agentes, instituições e organizações sociais. Tenta ser mais do que apenas um instrumento de representação de agentes discretos [Altaweel et al., 2002].

Na Figura 19, é apresentado um exemplo de um ecrã da interface gráfica do *Repast*. É possível observar as janelas distintas da aplicação: janela de gestão da temporização da simulação (canto superior esquerdo), janela da parametrização do modelo (canto superior direito), janela de apresentação de possíveis gráficos (canto inferior esquerdo) e janela de exposição / representação de todo o "mundo" da simulação em causa (canto inferior direito). A interface gráfica de apresentação e configuração da biblioteca *Repast* é semelhante a outras ferramentas utilizadas neste âmbito, como o *Swarm* e o *Mason* (descrito na secção 2.2.2.3).

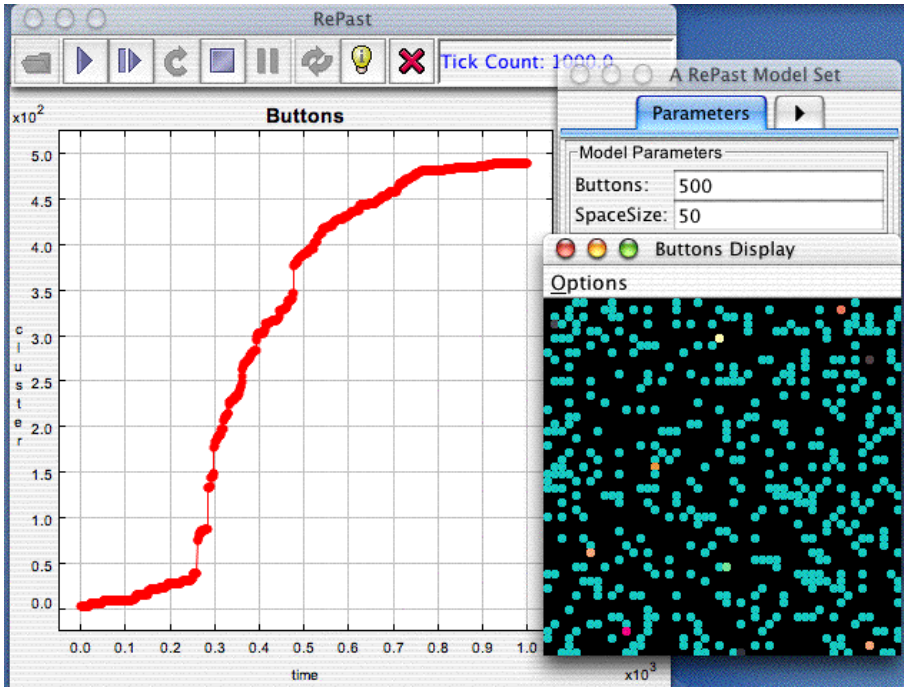


Figura 19. Impressão de Ecrã da Ferramenta de Simulação Repast

Existem três implementações distintas do *Repast*. Todas oferecem os mesmos serviços e diferenciam-se apenas na plataforma e linguagem sobre as quais foram desenvolvidas [Altaweel et al., 2002]:

1. *Repast J* (*Repast* para *Java*)
2. *Repast.Net* (*Repast* para *Microsoft.Net*)
3. *Repast Py* (*Repast* para *Python Scripting*)

2.2.2.3. *Mason*

Mason é uma aplicação-processo único de simulação, orientada a eventos discretos e simultaneamente uma ferramenta de visualização. É um sistema de utilização livre onde é fornecido o seu código fonte, à semelhança dos dois sistemas referidos nas secções anteriores. Foi desenvolvido conjuntamente pelo Departamento de Ciências Computacionais e pelo Centro para a Complexidade Social da Universidade George Mason localizada em Fairfax, Virgínia. A sigla *Mason* significa *Multi-Agent Simulator Of Neighborhoods* ou *Networks*, tendo também a sua origem no nome de George Mason [Luke et al., 2004].

Esta biblioteca foi construída com o intuito de ser rápida, ortogonal e minimalista. Não deriva de nenhuma outra biblioteca de simulação existente. É facilmente possível ao programador adicionar funcionalidades. Aos modelos *Mason* é possível agregar interfaces de manipulação e visualização 2D e 3D, úteis na percepção e observação da simulação. Tem a capacidade de armazenamento persistente de pontos de verificação durante a execução de uma determinada simulação. Possibilita assim interrupção de testes do modelo a qualquer momento, podendo-os migrar de plataforma [Luke et al., 2004].

O *Mason* é uma aplicação dividida em camadas diferentes (Figura 20), com a facilidade do acrescento de outros utilitários ou de associação a aplicações externas.

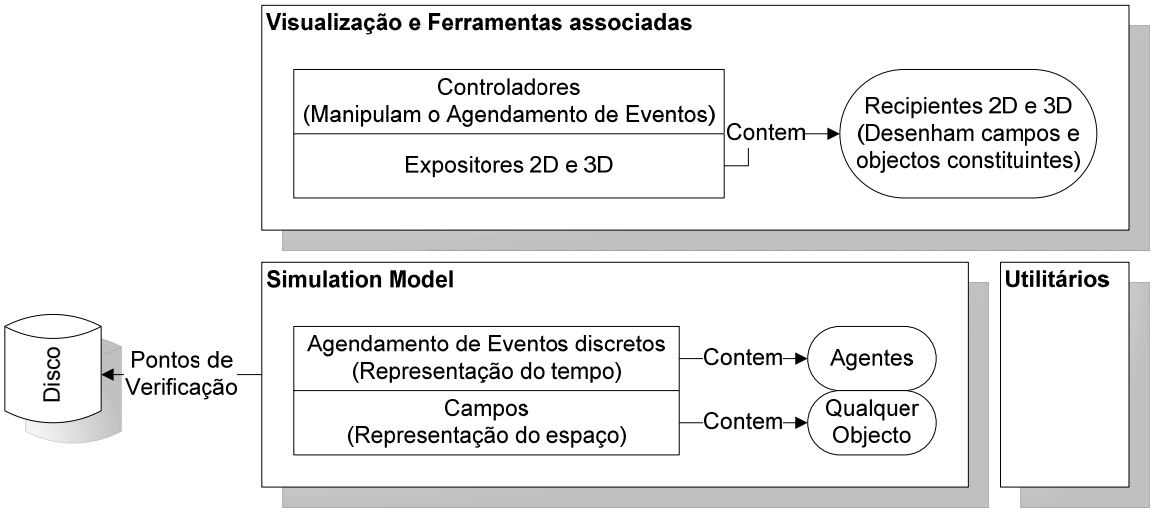


Figura 20. Modelo de elementos e camadas de visualização de MASON [Luke et al., 2004]

Ao contrário do *Repast* e do *Swarm*, este *software* não providencia ferramentas especificas de determinados domínios que são úteis a utilizadores não programadores, como por exemplo: gráficos, modelos físicos, entre outros. No entanto, como foi dito anteriormente, é uma biblioteca minimalista onde facilmente se adicionam novas características [Luke et al., 2004].

Depois do estudo destas ferramentas e bibliotecas aplicacionais de simulação, foi verificado que falta a noção de um padrão de utilização aquando da formalização e desenho do problema em causa. Por outras palavras, na simulação, falta um sistema de especificação e de redução às suas estruturas formais através de um processo de abstracção. Mais especificamente, falta a formalização de agentes, das suas interacções e do planificador (agendamento de eventos) numa simulação. De facto, existem linguagens de modelação que poderão auxiliar bastante nesta questão, linguagens que são descritas ao pormenor na próxima secção.

2.2.3. Linguagens de Modelação

Linguagem é um conjunto finito de símbolos combinados de acordo com regras gramaticais. Tem o propósito de comunicação e expressão de pensamentos, sendo já utilizada desde as primeiras civilizações humanas. A Figura 21 exemplifica a escrita hieroglífica que será provavelmente o mais antigo sistema organizado de linguagem escrita no mundo.

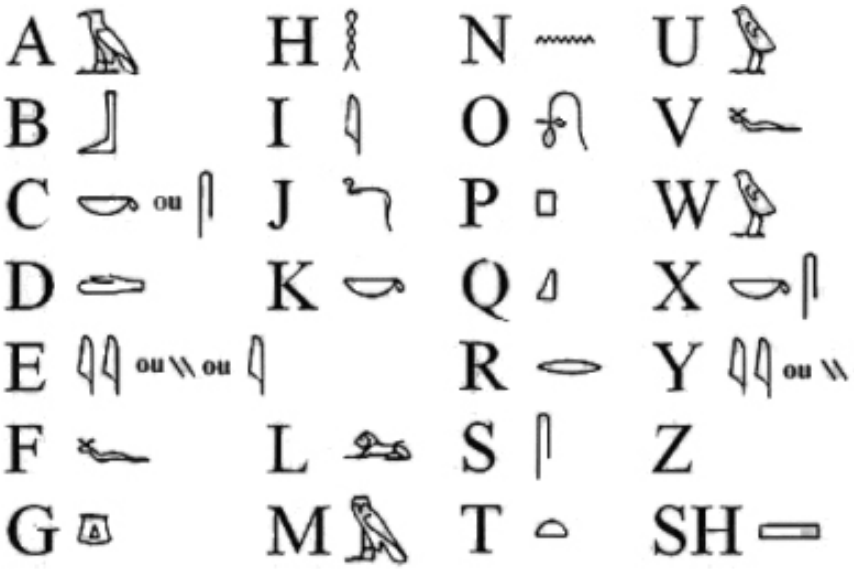


Figura 21. Hieróglifo – *Hierós* 'sagrado', e *Glyphós* 'escrita' [Stewart, 2006]

A análise realizada à estruturação de um sistema e, seguidamente, a especificação de um modelo desse próprio sistema é muito proveitosa. Isto porque representa, simplificadamente e com um certo grau de abstracção, o processo real. Dá, à partida, um amplo conhecimento do domínio em causa, deixando detalhes de implementação de lado. Para esta modelação, é necessário a utilização de linguagens que permitem visualizar o sistema sobre diferentes aspectos. Têm como objectivo final o entendimento e a organização do modelo em causa. Existem três exemplos de linguagens de modelação já muito explorados, em que os dois primeiros serão descritos com mais pormenor e o terceiro já foi discutido [Filgueira & Costa, 2004]:

- UML - *Unified Modeling Language* (Padrão)
- AORML - *Agent-Object Relationship Modeling Language* (Sistemas Multi-Agente)
- Redes de Petri (Sistemas Distribuídos)

2.2.3.1. UML – *Unified Modeling Language*

A linguagem de modelação UML tem origem em trabalhos publicados, já desde 1992, por Grady Booch, Jim Rumbaugh e Ivar Jacobson. Acaba por ser o resultado da fusão de vários sistemas de investigação: Booch (G. Booch), OMT (J. Rumbaugh) e OOSE (I. Jacobson). A primeira versão da linguagem surge em 1997. Entretanto foram criadas sucessivas versões contendo melhorias e características adicionais. É uma linguagem padrão (*standard*) para o

desenho de software, sendo actualmente apresentada pelo *Object Management Group* (*standard OMG*). É uma linguagem perfeitamente adaptável ao desenvolvimento aplicacional orientado a objectos [Louçã, 2007] [Kratz, 2000] [Crespo & Carvalho, 2005].

A utilização desta modelação UML permite observar globalmente a aplicação, tendo menos informação que o código fonte o que aumenta a legibilidade. Estrutura o sistema, não indo aos detalhes da implementação. O UML denota uma representação gráfica, o que aumenta a clareza do processo. Implementa vários tipos de ligações entre os elementos do modelo, nomeadamente a relação de generalização, dependência, realização, associação e mudança de estado. A linguagem UML é independente das ferramentas de modelação [Silva & Videira, 2005] [Crespo & Carvalho, 2005].

O UML apresenta tipos de diagramas diferentes. Cada um representa ou captura uma vista distinta do sistema a ser modelado, o que permite a observação deste sob perspectivas diferentes. Nesta linguagem existe o conceito de distinção entre a noção de diagrama e de modelo. Um diagrama é uma visão particular e parcial de certos elementos do modelo, detalhando a informação desse trecho do modelo. O modelo, por sua vez, contém todos os componentes do sistema e não tem informação da maneira como eles são expostos visualmente. Um distinto elemento tem apenas uma e só uma definição no modelo, mas poderá aparecer em múltiplos diagramas, apresentado de forma diferente [Kratz, 2000] [Filgueira & Costa, 2004].

Os diferentes diagramas existentes estão enumerados na próxima lista e exemplificados graficamente na Figura 22 [Louçã, 2007] [Crespo & Carvalho, 2005]:

- Modelação estrutural
    - Classes (... e objectos e pacotes): Organização das classes através do diagrama dos seus elementos (objectos, pacotes,...) com relações de inclusão, herança, entre outras.
    - Físico (Componentes e Configuração): Dependências entre componentes e respectiva configuração.
- Modelação comportamental
    - Caso de uso: Relações entre actores e "casos de uso".
    - Interacção (Sequência e Colaboração): Sequência temporal de interacção entre objectos e respectivas mensagens.
    - Estados: Sequência de estados pelos quais um processo ou objecto passam.
    - Actividade: Estados em que estão associados acções.

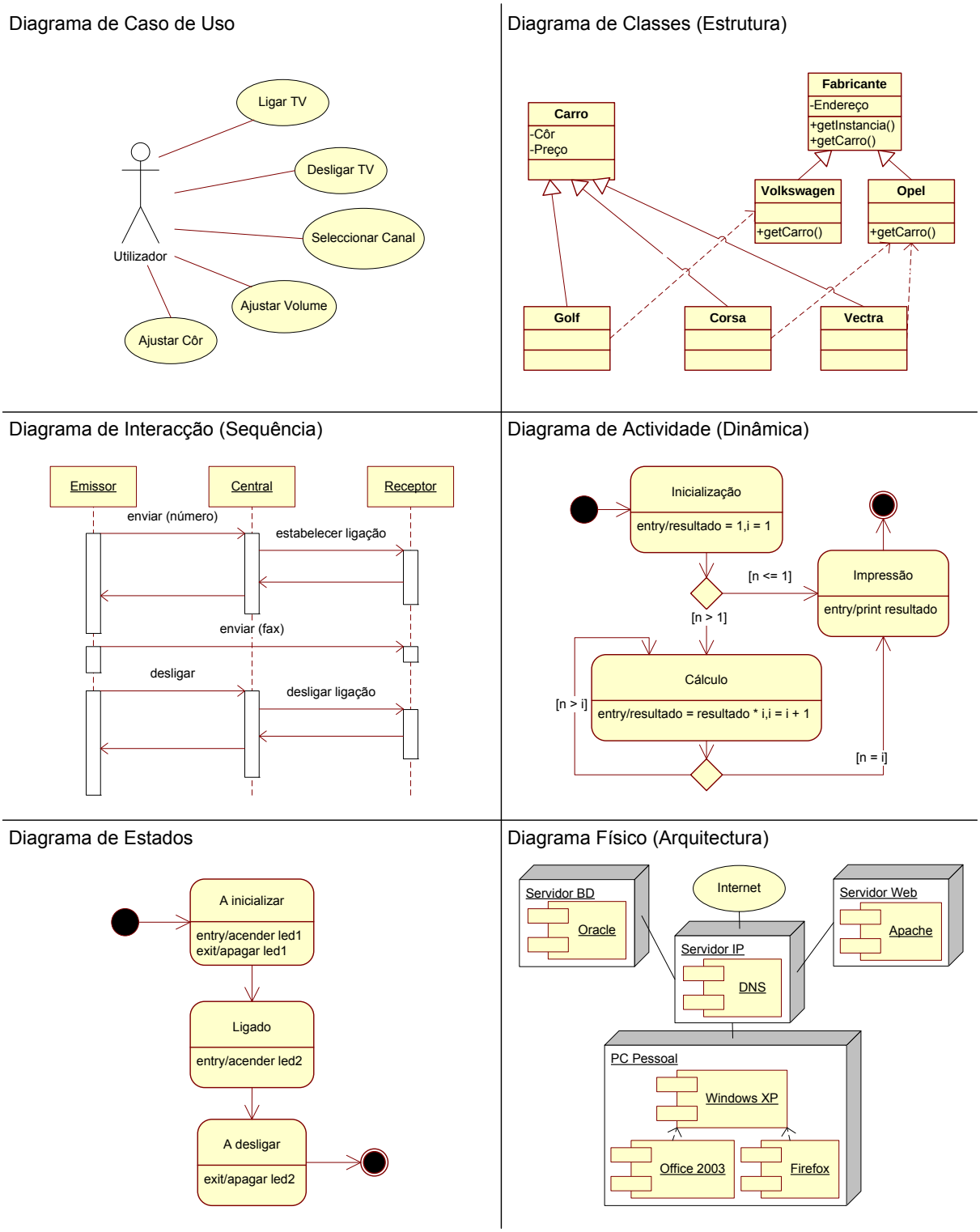


Figura 22. Diagramas de modelação UML

2.2.3.2. AORML – *Agent-Object Relationship Modeling Language*

Esta linguagem denominada AORML, *Agent-Object Relationship Markup / Modeling Language*, tem origem nos trabalhos realizados por Gerd Wagner, desde 2000, nomeadamente o artigo [Wagner, 2000]. Um diagrama nesta linguagem, orientado a agentes, tenta ser uma extensão ao diagrama de classes UML. No diagrama UML, não era possível retratar um agente correctamente, pois tornava-o como um objecto simples igual a todos os outros, sem actividade, sem acções. Esta extensão tenta suprimir essa limitação, pois tem como noção base a distinção entre agentes e objectos ordinários do ambiente que os rodeia. Por outras palavras, existe a distinção entre entidades activas e passivas num modelo de um sistema, o que possibilita a representação mais detalhada da entidade activa [Wagner, 2003] [Taveter & Wagner, 2001] [Nery et al., 2004].

Num modelo AORML, uma entidade poderá ser uma das seguintes opções: agente, objecto, acção, evento, compromisso e pedido. Apenas e só os agentes podem comunicar, actuar, cumprir e satisfazer pedidos, enquanto os objectos são entidades completamente passivas sem qualquer das capacidades dos agentes. Estas entidades agentes poderão desempenhar o papel de humanos ou instituições. Têm representações distintas para cada um dos casos, sendo que os agentes institucionais poderão ser grupos de entidades que actuam em prol de uma organização ou instituição [Brandão, 2005] [Wagner, 2003].

Existem dois tipos de modelos no AORML: o modelo externo e o modelo interno. O externo, também denominado de modelo de análise, delineia o sistema como um todo. Isto é, tem a perspectiva de alguém externo ao processo que presencia as iterações entre agentes e o ambiente que os rodeia. Neste modelo externo, as acções são simultaneamente eventos, e os pedidos são também compromissos. O tipo interno, também denominado de modelo de design, projecta o próprio agente. Corresponde à vista na primeira pessoa de um agente a ser modelado. É a visão de cada agente, como comunica com outros agentes, como reage a eventos, resumindo, como interage com o mundo [Wagner, 2003] [Nery et al., 2004].

A Figura 23 contém um exemplo de um diagrama externo AORML. O cenário é a modelação de um elevador, onde são descritas as acções que este realiza (Abrir Porta, Fechar Porta, entre outras), os eventos a que reage (Fecho de Porta, Chegada a Piso) e outros agentes ou objectos com quem interage. Os objectos passivos, como o "Poço", são representados por rectângulos e os agentes por rectângulos arredondados como o "Elevador" e o "Utilizador de

Elevador". As setas "Pedidos (Chamada ou Transporte)" no diagrama retractam "entidades pedidos" do agente "Utilizador de Elevador" ao agente "Elevador" [Wagner, 2003].

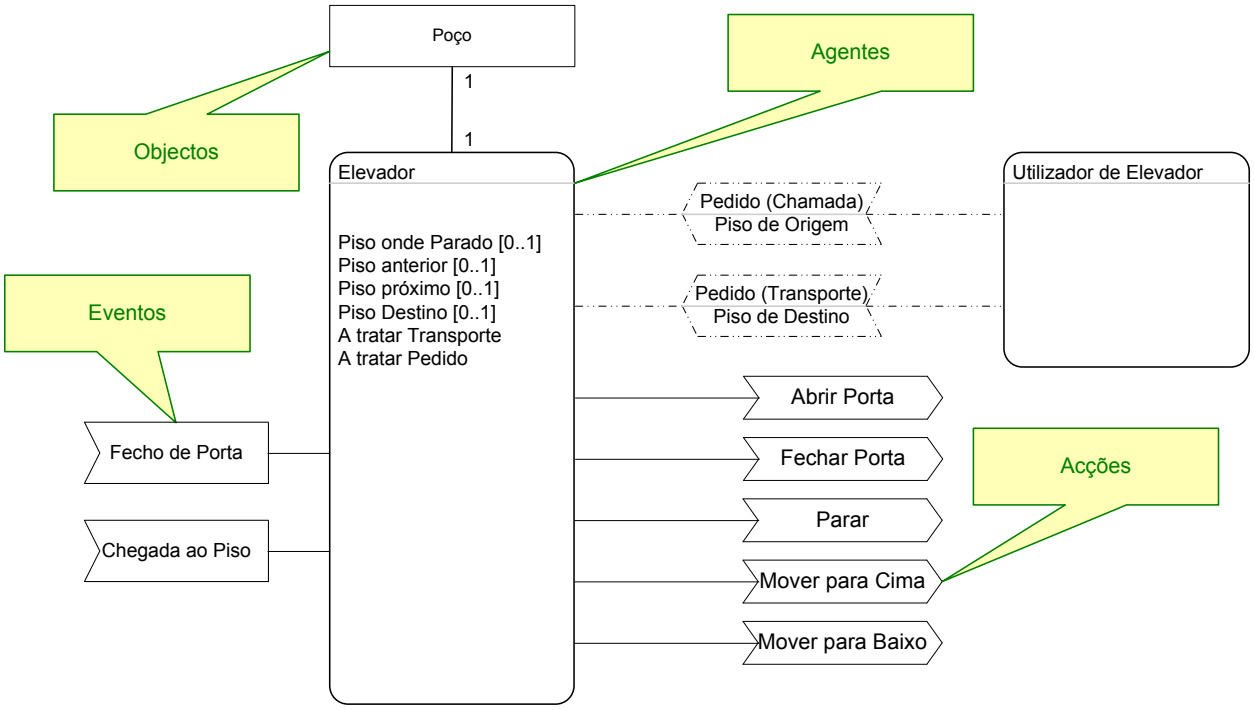


Figura 23. Diagrama de modelação AORML

2.2.4. Conclusão

Ao referir algumas linguagens de modelação existentes, bem como as ferramentas e bibliotecas aplicacionais de auxílio à simulação de sistemas multi-agente, ficou concluído que seria possível e benéfico a interligação destes dois conceitos. É desejável o desenho e formalização dos agentes e das suas interacções. Assim, na secção seguinte são descritos exemplos relacionados com esta ideia.

2.3. Modelos Redes de Petri em Simulação Baseada em Agentes

Esta secção apresenta três exemplos de projectos de sistemas multi-agente, onde existiu a utilização de Redes de Petri.

2.3.1. Exemplo 1 – *Colored Petri Net for a Multi-Agent Application*

Este trabalho intitulado *"Colored Petri Net for a Multi-Agent Application"* foi realizado por Danny Weyns e Tom Holvoet em 2002.

Neste exercício, foi realizada a modelação de uma aplicação multi-agente em Redes de Petri Coloridas e efectuada a combinação das experiências simulatórias com a modelação conceptual. A aplicação em causa, o *"Packet-World"*, consistiu num mundo virtual, onde existia um vasto número de pacotes espalhados que apresentavam cores diferentes. Os agentes deste mundo tinham o objectivo de recolher estes pacotes, armazenando-os no local com a cor correspondente. Os agentes tinham uma vista limitada do ambiente que os rodeava [Weyns & Holvoet, 2002].

Foi um tema de pesquisa proposto para a investigação social comportamental (estudos sociais) em simulação multi-agente. Tinha características ideais para análises e simulações: existia um mundo que tornava fácil a visualização de uma possível cooperação entre agentes, de uma possível partilha de informação entre agentes ou de uma possível competição entre agentes. A Figura 24 ilustra uma representação gráfica e um modelo do mundo *"Packet-World"* [Weyns & Holvoet, 2002].

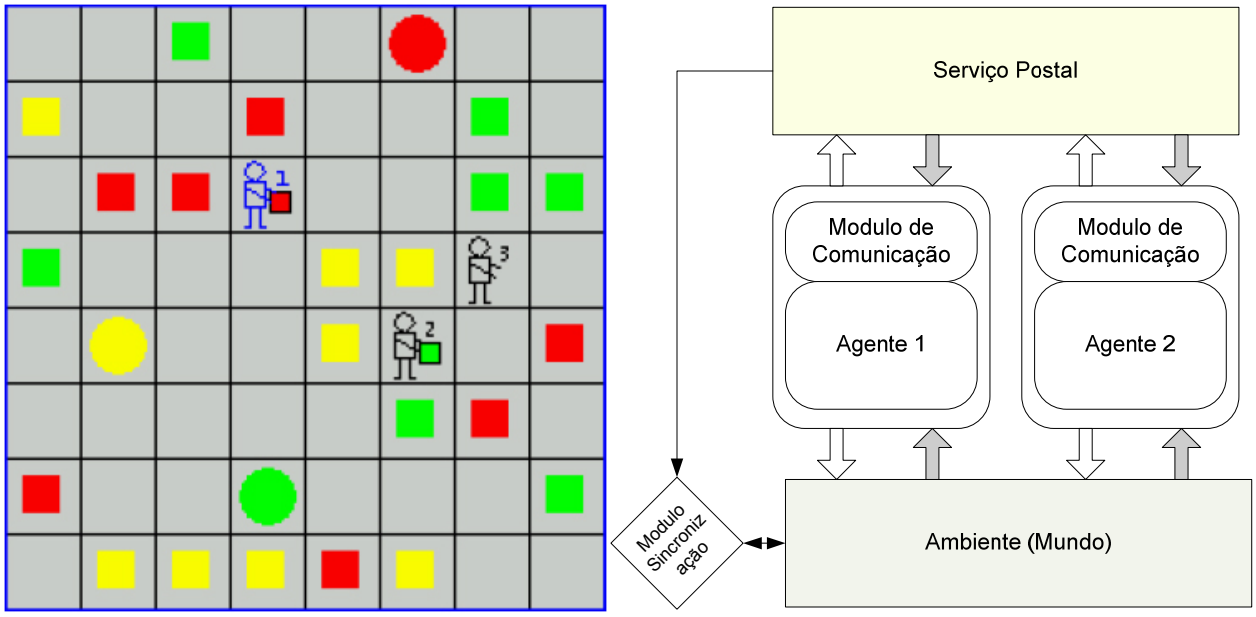


Figura 24. À esquerda – Representação gráfica do mundo, à direita – Modelo do problema [Weyns & Holvoet, 2002]

Este trabalho teve como objectivo principal o estudo de dois conceitos: o entender e o observar sociedades em simulação baseada em agentes e o explanar da modelação conceptual genérica de aplicações, em particular, aplicações multi-agente. A aproximação de modelação do problema em causa foi realizada de uma forma modular. Primeiro foi construído o modelo totalmente básico do *"Packet-World"*. De seguida, foi-se adicionando, à rede base, qualidades e características, ou seja, os ditos módulos. Estes novos módulos eram novos diagramas, onde apareceriam especificações de novas habilidades dos agentes; novas facetas do ambiente que os rodeava ou novas interligações entre as várias entidades existentes. Todos estes acrescentos foram efectuados de uma forma gradual. Os autores, para além do contexto de simulação baseada em agentes na área das ciências sociais, pretendiam generalizar esta aproximação modular de diagramas a qualquer outro contexto [Weyns & Holvoet, 2002].

Esta forma gradual da construção do modelo conceptual foi exemplificada para o *"Packet-World"*. Inicialmente foi apresentado uma primeira Rede de Petri Colorida base da aplicação. Consistia em agentes e a sua interacção com os objectivos passivos do ambiente. Seguidamente foi acrescentado a noção de coordenação entre estes mesmos agentes e, posteriormente foi adicionado o conceito de comunicação entre agentes ("comunicação é a base de organização social") e assim sucessiva ou progressivamente.

Neste projecto, foram tiradas as seguintes conclusões em relação à utilização de Redes de Petri Coloridas como ferramenta de modelação [Weyns & Holvoet, 2002]:

1. Redes de Petri têm tradição na análise e descrição de processos distribuídos e concorrentes.
2. Redes de Petri apresentam uma representação gráfica simples e intuitiva de processos complexos, permitindo também o estudo de dupla forma: a formal e a simulação.
3. Redes de Petri Coloridas são a melhor combinação das linguagens de programação com as Redes de Petri originais.

2.3.2. Exemplo 2 – *Modeling and Analysis of a Multi-Agent System Using Colored Petri Nets*

Esta dissertação com o título *"Modeling and Analysis of a Multi-Agent System Using Colored Petri Nets"* foi elaborada por Márcia Costa Miranda e Ângelo Perkusich em 1999.

Este trabalho, como o próprio título indica, apresentou a modelação de sistemas sociais baseados em agentes em Redes de Petri Coloridas. Para além da modelação, as Redes auxiliaram na análise e verificação do processo em causa. Este sistema estudado teve o nome de

"MATHEMA" e foi baseado em conceitos multi-agente. Consiste num ambiente de aprendizagem interactivo, tendo como principal finalidade a resolução de variados tipos de problemas matemáticos [Miranda & Perkusich, 1999].

Na arquitectura deste processo, cada agente foi definido de acordo com uma estrutura hierárquica. Existia um sistema social que era encarregado da promoção das interacções cooperativas ou comportamentos entre os agentes. Os agentes coadjuvam entre si para poder resolver o problema apresentado. Estes agentes interagiam de uma maneira cooperativista com o intuito de aperfeiçoar as actividades de aprendizagem / ensino num ambiente computacional interactivo [Miranda & Perkusich, 1999].

A Figura 25 é o diagrama da arquitectura funcional dos agentes, onde se verifica que a estrutura de um agente era composto por 3 módulos principais: Tutorial, Distribuição e Social [Miranda & Perkusich, 1999].

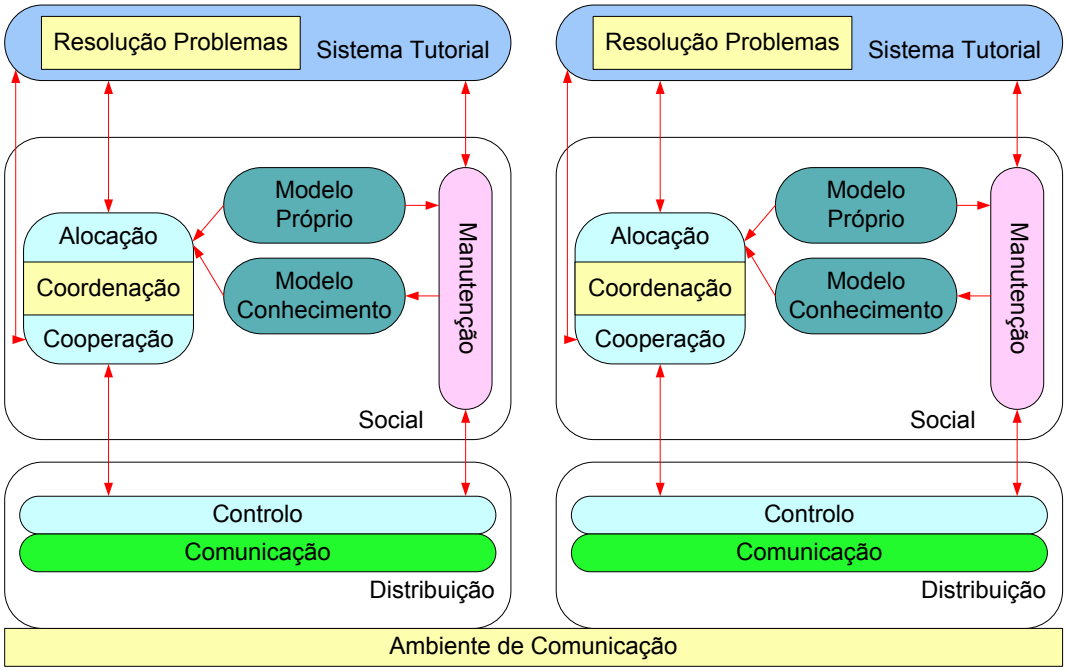


Figura 25. Diagrama Arquitectural dos Agentes [Miranda & Perkusich, 1999]

A componente de distribuição era responsável pela recepção e envio de mensagens através do ambiente de comunicação; a componente tutorial encarregava-se das interacções entre o utilizador humano e o agente; e a componente social incumbia-se da coordenação das actividades cooperativas com outros agentes. Cada um destes componentes continha sub-módulos interligados entre si. Por exemplo, os sub-módulos Alocação, Coordenação e

Cooperação no componente Social, como mostra a Figura 25. Neste trabalho, a aproximação de modelação em Redes de Petri foi dividir e conquistar. Foram criados sub-modelos para cada sub-módulo existente. Um dos exemplos de Redes de Petri apresentados neste projecto foi o relativo à parte de Alocação da componente Social. O diagrama é visível na Figura 26 [Miranda & Perkusich, 1999].

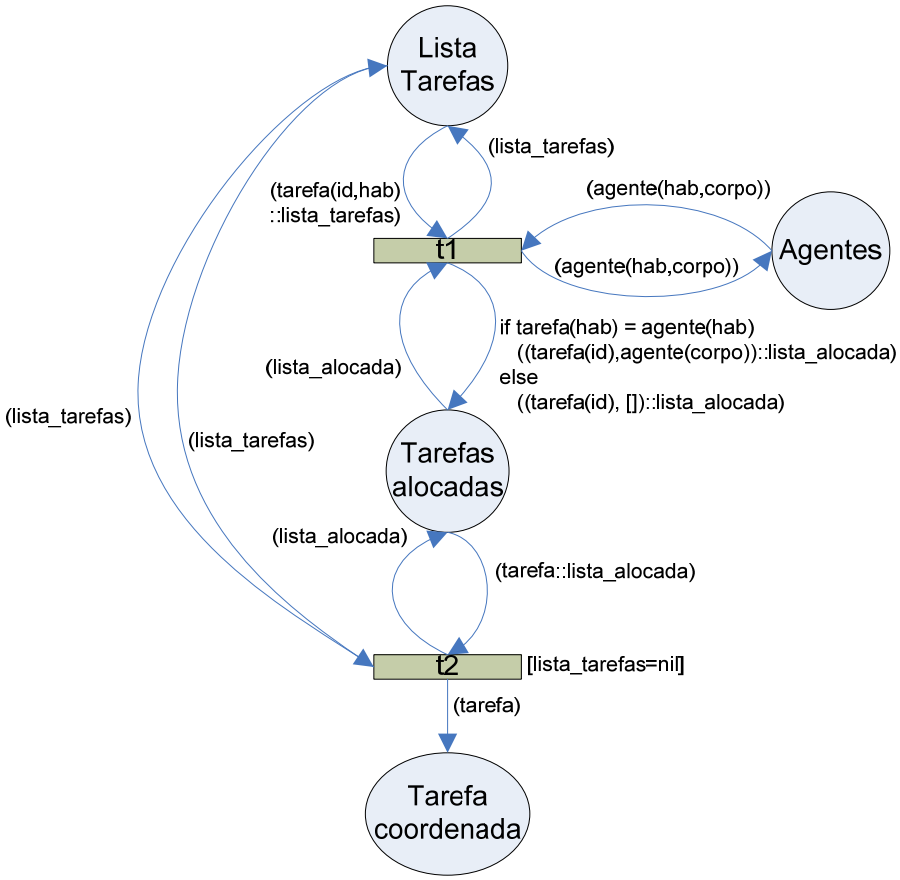


Figura 26. Módulo de alocação de tarefas – Rede de Petri Colorida [Miranda & Perkusich, 1999]

Neste módulo e no respectivo modelo representativo, são recebidas tarefas decompostas (representadas por marcas) inicialmente colocadas no lugar "Lista Tarefas". Depois, na lista de agentes do lugar "Agentes" (agentes igualmente representados por marcas), caso exista um agente capaz que resolver a determinada tarefa, esta passa a ser alocada e por fim coordenada (lugares "Tarefas alocadas" e "Tarefa coordenada" respectivamente) [Miranda & Perkusich, 1999].

As Redes de Petri auxiliaram também na análise das várias simulações efectuadas ao sistema em causa. Essas análises foram maioritariamente realizadas através de árvores de

cobertura com a finalidade de estudar propriedades e comportamentos do sistema. Consideraram diversos cenários expressados por diferentes marcações iniciais no modelo (equivalente a diferentes tipos e números de tarefas submetidas ao "MATHEMA"). Na dissertação, foram apresentados vários exemplos que descreviam a vivacidade das redes resultantes. Foram testadas se as redes eram vivas ou não, se existiam bloqueios (*deadlocks*), se existiam transições mortas, entre outras propriedades [Miranda & Perkusich, 1999].

Na elaboração das Redes de Petri Coloridas foi utilizada uma ferramenta auxiliar designada DesignjCPN [Cherkasova et al., 1993]. No final desta dissertação, foi concluído que a aplicação de Redes de Petri neste sistema de simulação baseada em agentes foi uma boa escolha, pelas seguintes razões [Miranda & Perkusich, 1999]:

1. Permitem a descrição de características como a concorrência e paralelismo e promovem a representação de tipo de dados complexos (redes coloridas).
2. Existem ferramentas que auxiliam no estudo, edição e confirmação de sintaxe das Redes, apresentando ambientes gráficos amigáveis.
3. Os modelos permitem detectar antecipadamente problemas de especificação do processo.

2.3.3. Exemplo 3 – *Socionic Multi-Agent Systems Based on Reflexive Petri Nets and Theories of Social Self-Organisation*

A exposição com a designação *"Socionic Multi-Agent Systems Based on Reflexive Petri Nets and Theories of Social Self-Organisation"* foi produzida em 2007 por Michael Köhler, Roman Langer, Rolf von Lüde, Daniel Moldt, Heiko Rölke e Rüdiger Valk.

O principal objectivo do trabalho foi construir e modelar uma teoria social com a capacidade de explicar diferenciadas estruturas e processos auto-organizados: a intitulada Teoria de Auto Organização Social (TSSO - *theory of social self-organisation*). Esta teoria foi o resultado do estudo e análise de semelhanças de várias outras teorias. Este resultado tentava provar que existiam mecanismos instintivos básicos e despercebidos de auto-organização e constante reestruturação. Estudava o rearranjo de estruturas sociais, contemplando todas as acções que estas unidades sociais geravam, estabilizavam ou modificavam. Neste trabalho, o contexto social estudado é o educacional e as suas instituições (universidades, professores, …). Esta hipótese teórica apresentava as seguintes propostas [Köhler et al., 2007]:

- Existe uma dinâmica de constituição numa unidade social, onde esta se recria e reproduz automaticamente.
- Existe uma dinâmica de estruturação numa unidade social, onde se verificam problemas na interacção entre duas unidades sociais.

Para a prova destas teorias, foi construído um sistema de arquitectura multi-agente que tentava adoptar as características naturais de contextos sociais, ao qual foi dado o nome de SONAR. Nesta elaboração do sistema, foi utilizado o seguinte processo recursivo ilustrado e auto-explicado na Figura 27. É formulada a teoria, é modelada por Redes de Petri e é avaliada. Consoante os resultados da avaliação é ou não melhorada (recursão). No final, era implementada a aplicação multi-agente SONAR [Köhler et al., 2007].

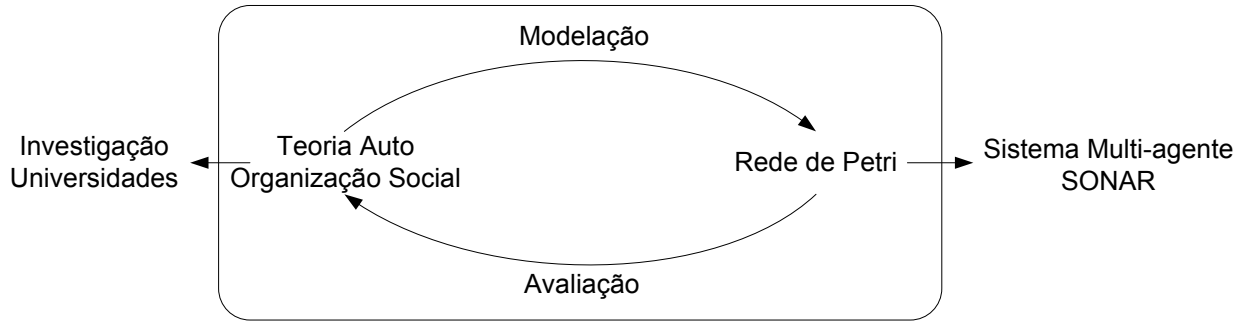


Figura 27. Processo utilizado na aproximação ao problema [Köhler et al., 2007]

Um dos objectivos deste projecto foi o desenvolvimento de uma aplicação baseada em agentes (SONAR) que agrupasse as ideias resultantes das teorias sociais. Os agentes continham atributos como a mobilidade, adaptabilidade, cooperação e autonomia. Para isto foi realizada modelação multi-agente usando a arquitectura MULAN [Köhler et al., 2001] [Duvigneau et al., 2003], modelação essa que é baseada em sub-Redes de Petri dentro de Redes de Petri (Redes de Petri Hierárquicas / Redes de Petri de Referência).

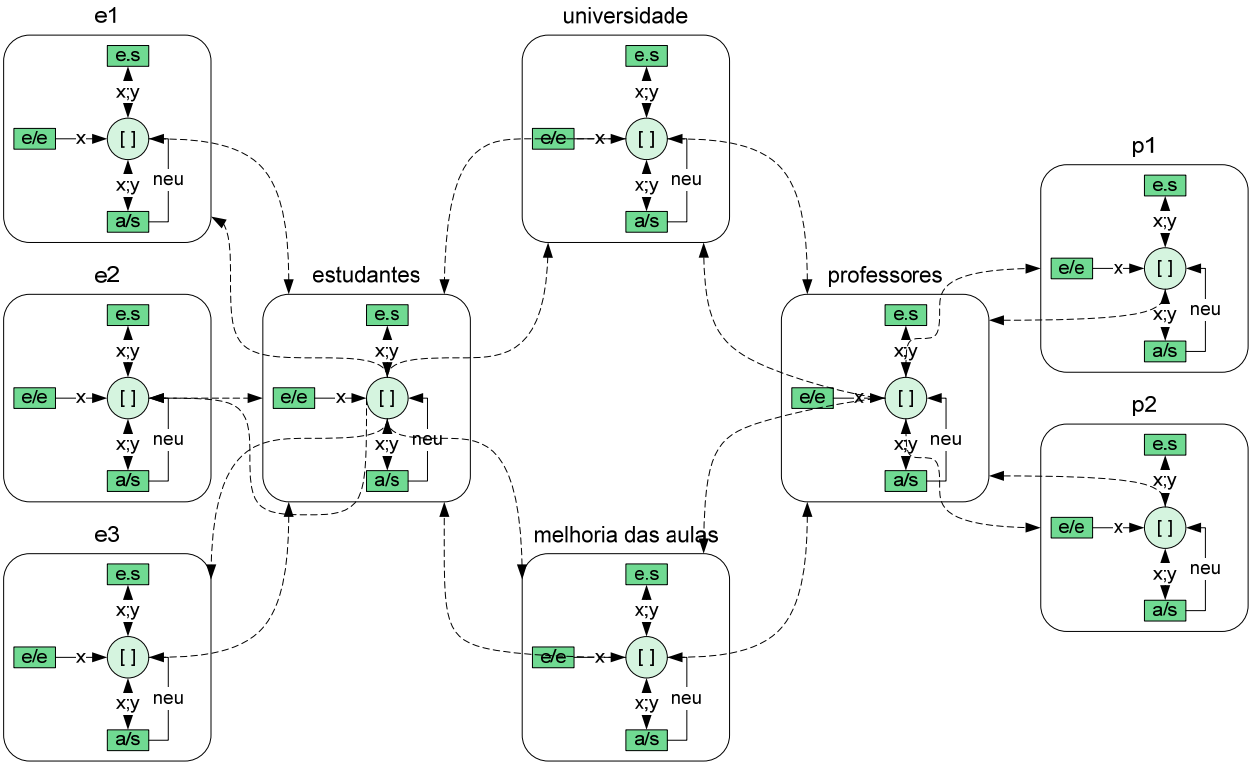


Figura 28. Rede de Agentes do sistema SONAR

Um agente do processo SONAR não é obrigatoriamente uma entidade ou actor, poderá ser também um sistema de regras / capacidades que dêem resultado a actos sociais. Na Figura 28, é ilustrado um modelo do SONAR, onde cada sub-rede representa um agente. Estas sub-redes (agentes: professores, alunos, universidade) interligam-se entre si [Köhler et al., 2007].

No final do projecto foram enumerados os principais motivos pelos quais foram escolhidas as Redes de Petri na modelação deste sistema denominado SONAR [Köhler et al., 2007]:

1. São executáveis, isto é, é possível processá-las com uma determinada marcação inicial e logo obter resultados.
2. É possível a análise prévia estrutural bem como comportamental da aplicação em causa.
3. Contêm uma representação visual gráfica, o que ajuda à sua percepção.
4. Existe a noção de concorrência bem como as características relacionadas: causalidade, paralelismo, alternativas, não determinista, entre outras.

5. Abrange o formalismo da linguagem de modelação UML. Existe a representação de entidades passivas e activas e é ainda possível especificar tipos de dados estruturados, a noção de tempo e de orientação a objectos.

2.3.4. Conclusão

Foram estudados vários textos relacionados com a utilização de Redes de Petri como auxilio ao desenvolvimento de simulações multi-agente. Três desses projectos foram aqui relatados. Depois deste estudo, foi verificado que o que falta efectivar é uma possível normalização e generalização na concepção (desenho) do modelo nos sistemas de simulação multi-agente. Esta dissertação tenta colmatar esta falta.

Resumidamente, esta dissertação propõe a normalização da modelação por Redes de Petri como consequência (resultados, produto) de simulações efectuadas em aplicações multi-agente. Considera as Redes de Petri como ferramenta auxiliar às interpretações dos resultados dessas mesmas simulações. Este assunto será aprofundado nos capítulos subsequentes.

3. PROPOSTA

Este capítulo descreve a proposta desta dissertação, que consiste na utilização de Redes de Petri como ferramenta para a visualização de simulações multi-agente. Para concretizar esta ideia foi concebido um acrescento à biblioteca aplicacional de simulação multi-agente *Repast*.

3.1. Extensão ao *Repast* para desenho em tempo real de Redes de Petri

A plataforma *Repast* contém interfaces gráficas que permitem representar resultados de simulações, como as janelas de apresentação de gráficos e representação de todo o "mundo" da simulação em causa. Estas representações gráficas nativas do *Repast* ajudam à visualização da simulação.

Foi proposta uma nova apresentação gráfica em representação de simulações. Um modelo gráfico que permite a descrição de características como a concorrência e o paralelismo do sistema simulado e que possibilita a análise e descrição de sistemas distribuídos. Foi, então, adicionado um novo módulo gráfico ao *Repast*: Redes de Petri como uma nova consequência gráfica de simulações. Tem como intuito auxiliar os potenciais investigadores, acrescentando uma nova visão gráfica dos resultados obtidos.

A implementação desta extensão ao *Repast* correspondeu a três passos principais. O primeiro passo foi identificar um exemplo simples de uma simulação baseada em agentes. Um exemplo onde, no seu estudo, seria benéfico a geração da respectiva Rede de Petri, ilustrando simultaneamente as capacidades de análise da mesma Rede. O segundo passo foi a definição da Interface Aplicacional de Programação (API) desta novo módulo da biblioteca, isto é, determinar a forma de comunicação entre o programador de uma simulação e a respectiva biblioteca. Finalmente, o último passo consistiu em implementar toda a apresentação e interface gráfica dos novos resultados em forma de Redes de Petri.

3.1.1. Problema proposto

O modelo utilizado foi o já famoso problema do Jantar de Filósofos [Louçã, 2006]. Este modelo, já referido no capítulo do Estado da Arte, é um exemplo elucidativo de concorrência e

sincronização de sistemas multi-processo. É indicado para a representação em modelação de Redes de Petri.

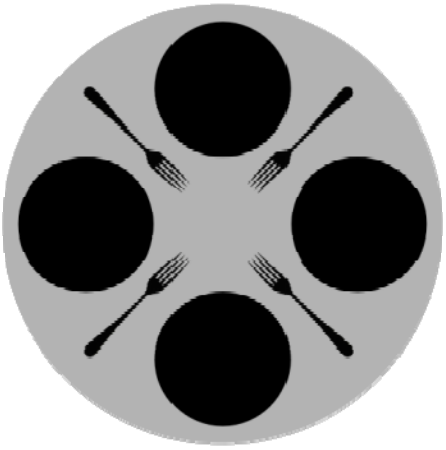


Figura 29. Distribuição de talheres na mesa do Jantar de Filósofos

O problema dos filósofos foi modelado em *Repast*, usando a aproximação da simulação baseada em agentes. Neste caso o agente foi o próprio filósofo. O filósofo apresentava uma sequência de acções.

Neste problema, os filósofos estão a jantar sentados à volta de uma mesa circular, tendo um prato de esparguete à sua frente com um garfo de cada lado (um garfo por filósofo). A Figura 29 exibe a distribuição dos talheres na mesa. As acções do filósofo são comer ou pensar. Cada filósofo necessita de dois garfos para comer, não pegando em nenhum, caso não estejam ambos livres.

Nesta simulação, a mudança de atitude do agente é determinada aleatoriamente, isto é, cada filósofo fica ou deixa de ficar com fome aleatoriamente. De seguida são apresentados os modelos AORML que ajudam a explicar a simulação Jantar de Filósofos (Figuras 30 e 31).

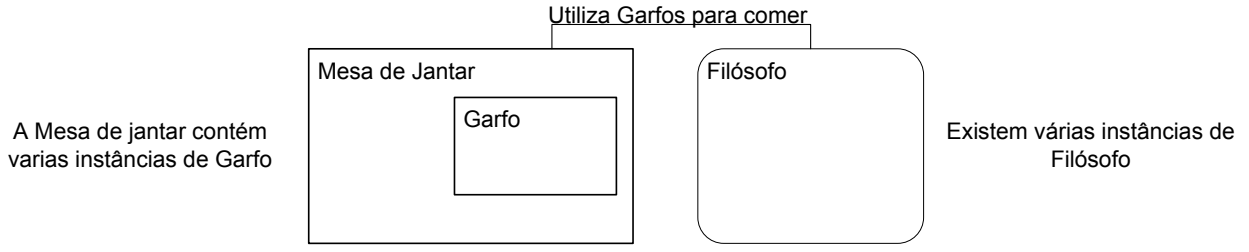


Figura 30. Diagramas AORML de entidades envolvidas

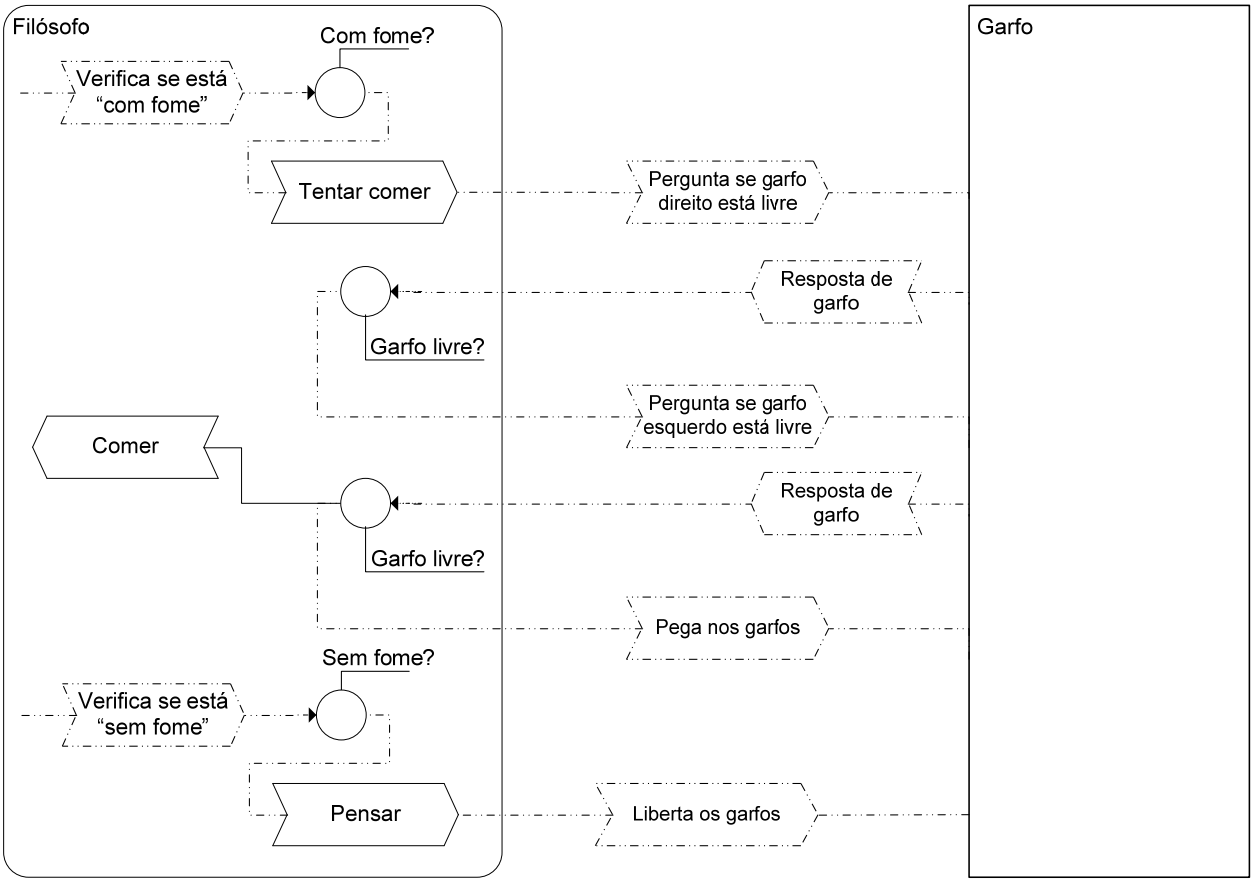


Figura 31. Diagrama AORML sequencial de interacções do agente

Para a implementação da simulação deste problema em *Repast*, foram desenvolvidas três entidades (classes):

1. Mesa de jantar (`DiningTable`) – representa o mundo da simulação que contém todas as entidades que nela participam, sejam agentes com as respectivas acções ou objectos estáticos. É a mesa de jantar onde os filósofos se sentam para comer ou pensar. É a implementação do modelo da simulação (`SimModelImpl`). É responsável pelo agendamento dos eventos dos agentes existentes no seu contexto. Por outras palavras, dirige a invocação das acções de todos os filósofos à volta da mesa. Nesta simulação existem os seguintes parâmetros: número de filósofos existentes na mesa; intervalo de tempo definido entre as acções (comer ou pensar) a executar pelos filósofos; indicador da existência de um especial filósofo glutão (conceito a explicar posteriormente no capítulo de avaliação da proposta). A Figura 32 apresenta a janela da parametrização do modelo.

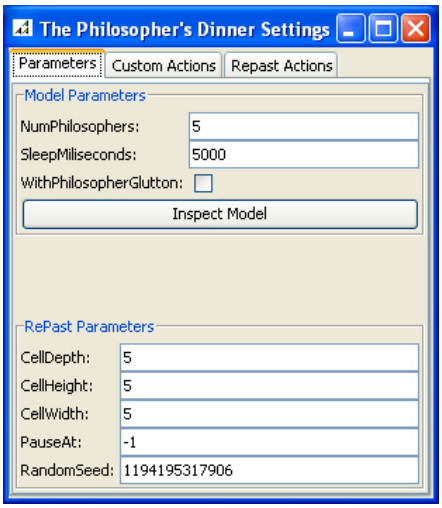


Figura 32. Janela de parametrização do modelo Repast

2. Filósofo (`Philosopher`) – É o agente da simulação. Disponibiliza o método público *"doAction"* que é responsável por executar as respectivas acções (*"eat"*; *"think"*; *"tryToEat"*). É constituído pelo seu estado actual (a comer; a pensar; com fome) e pelos garfos a que tem acesso (esquerdo e direito). Tem associação com os objectos estáticos garfos, usando-os quando necessário. A Figura 33 exibe a estrutura de classes, o que ajuda a perceber a organização da aplicação realizada. O agente é apresentado graficamente com os ícones:  - a pensar;  - a comer;  - com fome; (visível na Figura 34).

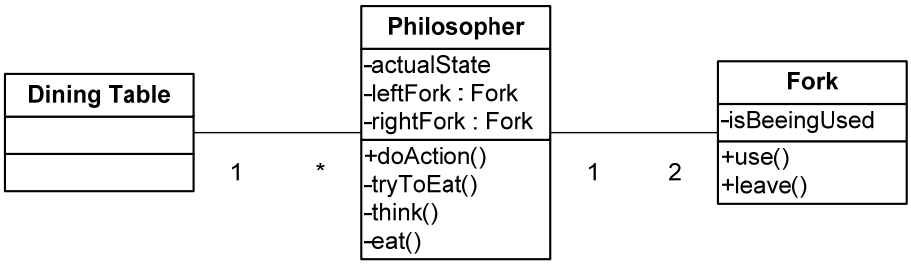


Figura 33. Estrutura de classes da simulação

3. Garfo (`Fork`) – A entidade estática garfo. Representa um garfo na mesa que poderá ser utilizado pelos agentes filósofos. Assim como o filósofo, o garfo contém um atributo a reflectir o seu estado (*isBeeingUsed* – indica se está a ser utilizado por alguém). Contém os

métodos públicos *"use"* e *"leave"* que permitem aos respectivos filósofos a sua utilização quando necessários. O objecto é representado com o ícone  (visível na Figura 34).

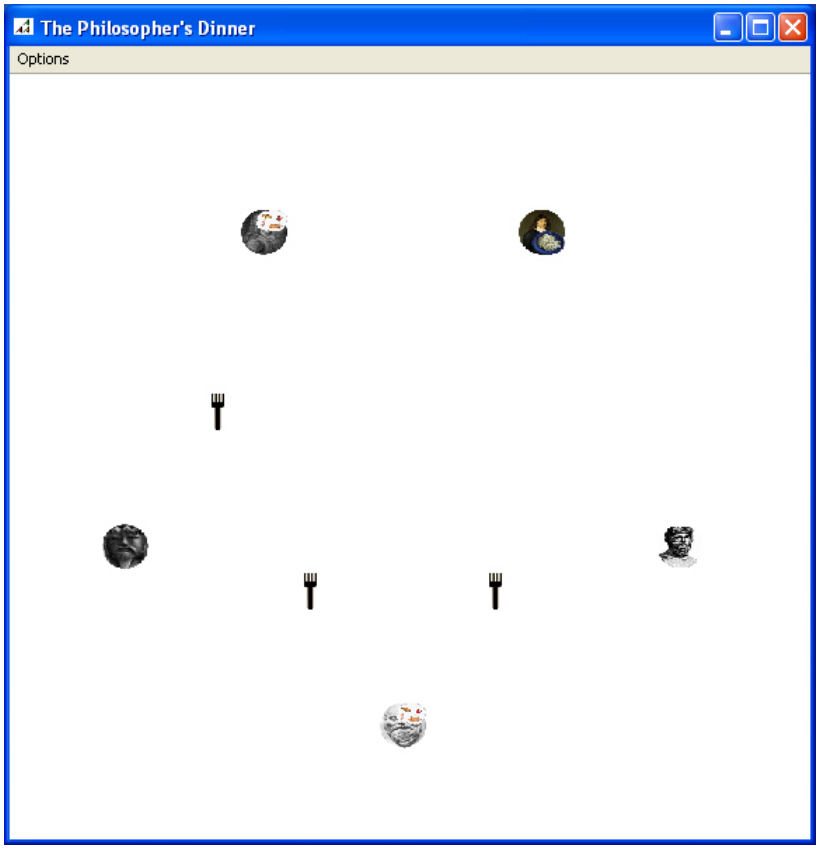


Figura 34. Janela de representação gráfica da simulação

O desenvolvimento da simulação do Jantar de Filósofos foi finalizada e testada. A simulação invocava e implementava as interfaces nativas da biblioteca *Repast*. Foi verificado que a aplicação apresentava o comportamento esperado do problema.

Foi criada uma boa base e um exemplo útil para o desenvolvimento da aplicação proposta por esta dissertação: o novo módulo da biblioteca de desenho de Redes de Petri em tempo real. As secções seguintes descrevem a realização da proposta, tendo como base este exemplo do Jantar de Filósofos.

3.1.2. Interface Aplicacional de Programação (API)

A API é o elo de ligação entre o módulo de Redes de Petri e o potencial programador de uma simulação *Repast*. Para a especificação da Interface Aplicacional de Programação foram consideradas as seguintes características:

1. Uma interface aplicacional de utilização simples e auto-explicativa.
2. Uma API que recolhesse toda a informação essencial e necessária para a geração das Redes de Petri correspondentes.
3. A característica mais importante: A utilização da API implicar um mínimo de alterações ao código fonte do cliente programador, mantendo a característica de recolha de informação necessária.

A interface, para a geração Redes de Petri em tempo real, tem de recolher dois tipos de informação:

- Informação estrutural: Dados relativos às acções e estados evidenciados pelos agentes envolvidos na simulação. Estes conteúdos dão origem a todos os nós de uma Rede de Petri, as transições e os lugares, reflectindo a sequência de eventos dos agentes.
- Informação temporal: Dados temporais relativos aos momentos em que as acções (eventos) são despoletadas nos agentes existentes na simulação. Esta informação é essencial para a actualização dos modelos de Petri em tempo real, criando o dinamismo na rede sincronizado com o decorrer da simulação.

Em seguida, são enumerados os passos que o programador necessita de realizar para dar a conhecer ao módulo de Redes de Petri a estrutura comportamental da sua simulação (informação estrutural):

1. Criar uma instância da classe `PetriNetModel`, à semelhança da utilização de qualquer classe do tipo *display* do *Repast*. No construtor desta classe, é parametrizado a lista de agentes envolvidos na simulação em causa. No exemplo utilizado, é criada a instância com a listagem de todos os filósofos do jantar.
2. Garantir que os elementos que interagem na simulação implementam a interface `PetriElement`. O próprio passo anterior impõe esta regra, uma vez que o parâmetro "lista de agentes", do construtor da classe `PetriNetModel`, é uma lista de objectos que implementam esta interface `PetriElement`. A definição da interface é simples:

```
// interface that must be implemented by the entities of the simulation
that are used by the petri net model
public interface PetriElement {
   String identifier();
   String description();
   PetriElement[] relatedElements();
}
```

O elemento `identifier` constitui o identificador único do componente na rede. O elemento `description` é uma breve descrição desse mesmo componente. A lista `relatedElements` contém outros possíveis elementos da rede associados a este. Para explicar melhor esta interface, é apresentado o código fonte do filósofo, de fácil programação, que implementa estas características da interface `PetriElement`. O identificador de um filósofo é, por exemplo, *"p1"*, a sua descrição *"philosopher 1"* e os elementos de rede relacionados consigo são os seus garfos esquerdo e direito. São os objectos estáticos da simulação que usa para "comer".

```
// identifier for PetriElement -> must be unique
public String identifier() {
   return "p" + index;
}

// description for PetriElement, used in legend
public String description() {
   return "philosopher " + index;
}

// returns the related petri Elements for PetriElement (forks)
public PetriElement[] relatedElements() {
   return new PetriElement[]{ leftFork, rightFork };
}
```

3. Neste último passo, o programador da simulação necessita de utilizar anotações Java (esta tecnologia está descrita no Anexo A. Anotações Java). Foi concluído que a obtenção da informação necessária à construção da rede residiu na ideia da aplicação de Anotações Java. Fornece ao programador a possibilidade de adicionar semântica de estados e transições de

estados sobre as próprias entidades da simulação *Repast*. O programador anota o seu código fonte sem necessidade de alterar a sua parte lógica e comportamental. Foram especificadas duas anotações possíveis de aplicação:

```
//Annotation used by the developers to indicate the initial state of the
simulation entities.
@Retention(RetentionPolicy.RUNTIME)
public @interface InitialState {
   String name();
}

// Annotation used by the developers to represent actions made by the
simulation entities.
@Retention(RetentionPolicy.RUNTIME)
public @interface Action {
   String[] originStates();
   String[] destinyStates();
   String name();
}
```

A primeira anotação `InitialState` serve para indicar qual o nome do estado inicial da entidade em causa. Esta anotação deverá ser utilizada no construtor da classe respectiva. A segunda anotação `Action` é utilizada na decoração dos métodos das classes de entidades da simulação, onde ocorram acções que mudem o estado do próprio objecto. São parametrizados os estados originários da acção, os estados resultantes da acção e o nome desta. É acrescentado o exemplo da classe do filósofo para melhor entender o uso destas anotações:

```
// constructor
@InitialState(name = "THINKING")
public Philosopher(int philosopherIndex, Point p) {
   index = philosopherIndex;
   actualPoint = p;
   actualState = State.THINKING;
   imageThink = new ImageIcon("images/think"+index%5+".gif").getImage();
   imageEat = new ImageIcon("images/eat"+index%5+".gif").getImage();
   imageHungry = new ImageIcon("images/hungry"+index%5+".gif").getImage();
}
```

```
// action "think" - leave forks and think
@Action(originStates={"EATING"},destinyStates={"THINKING"},name="ToTHINK")
private void think() {
   leftFork.leave();
   rightFork.leave();
   actualState = State.THINKING;
}

// action "eat" - take forks and eat
@Action(originStates={"THINKING"},destinyStates={"EATING"},name= "ToEAT")
private void eat() {
   rightFork.use();
   leftFork.use();
   actualState = State.EATING;
}
```

Neste exemplo, é observado que o filósofo tem como estado inicial o denominado `"THINKING"` definido na anotação no seu construtor. Passa para o estado `"EATING"` pela a acção `"ToEAT"`, voltando ao estado `"THINKING"` pela acção `"ToTHINK"` nos métodos `eat()` e `think()` respectivamente.

Depois da definição do modo de comunicação com o programador (API), foi desenvolvida a forma de instanciação das estruturas internas, do novo módulo de geração de Redes, que caracterizam os elementos da Rede de Petri. Com base nos parâmetros de entrada da classe `PetriNetModel` e a informação anotada, foi implementada a seguinte aproximação: No construtor da classe foi percorrido o parâmetro inicial "listagem de agentes `PetriElement`" e respectivos `relatedElements`. Os agentes foram inspeccionados um a um, à procura das referidas anotações (utilizando a API Java de *Reflection* - java.lang.reflect). A partir destes dados de estado e de mudança de estado encontrados, foi efectuado o cruzamento desta informação e criada toda a Rede de Petri composta pelos lugares e transições, a simbolizar os estados e acções respectivamente.

Para a recolha de dados temporais da ocorrência das acções dos agentes da simulação (informação temporal) foi utilizado uma outra tecnologia. Com o objectivo de animar, em tempo real, a Rede de Petri gerada, foi aplicada a programação orientada para aspectos (esta tecnologia está descrita no Anexo B. Programação Orientada para Aspectos). Foi usada a *framework*

AspectJ que é a implementação da orientação para aspectos em Java. Apenas foi necessário o desenvolvimento do aspecto `ActionObserver`:

```
// aspect to observe the simulation, that checks the invocations of actions by
the simulation entities
public aspect ActionObserver {

	// pointcut (whenever the program execution reaches it) - calling of
	methods that have the annotation @Action
	pointcut callAction(PetriElement petriElement, Action action) :
	call(@Action * *.*(..)) && target(petriElement) && @annotation(action);

	// before the pointcut, the petri net should be notified about the
	changes made by the action
	before(PetriElement petriElement,Action action) :
	callAction(petriElement, action) {
		PetriNetModel.actionTrigger(petriElement, action.name());
	}

}
```

O aspecto `ActionObserver` actua como um observador de mudança de estados de outros objectos. Implementa um esquema de notificações automáticas, para que, quando um estado de um objecto monitorizado é alterado, os nós da Rede de Petri dependente sejam automaticamente notificados. Mais concretamente, assim que qualquer método anotado por `@Action` é invocado durante a simulação *Repast* (de uma classe de interface `PetriElement`), o modelo de Petri é informado com o nome da acção invocada. A cada acção corresponde uma transição na rede. A transição é então disparada, provocando a mudança do número de marcas nos respectivos lugares de entrada e saída.

No exemplo concreto do problema dos filósofos, a chamada dos métodos `eat()` e `think()` na classe agente `Philosopher` provocam a comunicação de notificações ao modelo da rede. Isto porque estes métodos contêm a anotação `@Action` e a classe `Philosopher` implementa a interface `PetriElement` e que vai de encontro à especificação do *pointcut* do aspecto `ActionObserver` (…`call(@Action * *.*(..)) && target(petriElement)` …).

Ao contrário da recolha da informação estrutural por parte da API, a recolha de informação temporal com a utilização de aspectos é completamente transparente para o programador *Repast*. O programador cliente não necessita de realizar qualquer alteração ao seu código fonte de simulação, nem se apercebe da existência deste mecanismo orientado a aspectos.

A Figura 35 ilustra, de uma forma simplificada, o sistema idealizado e desenvolvido na Interface Aplicacional de Programação, descrito nesta secção.

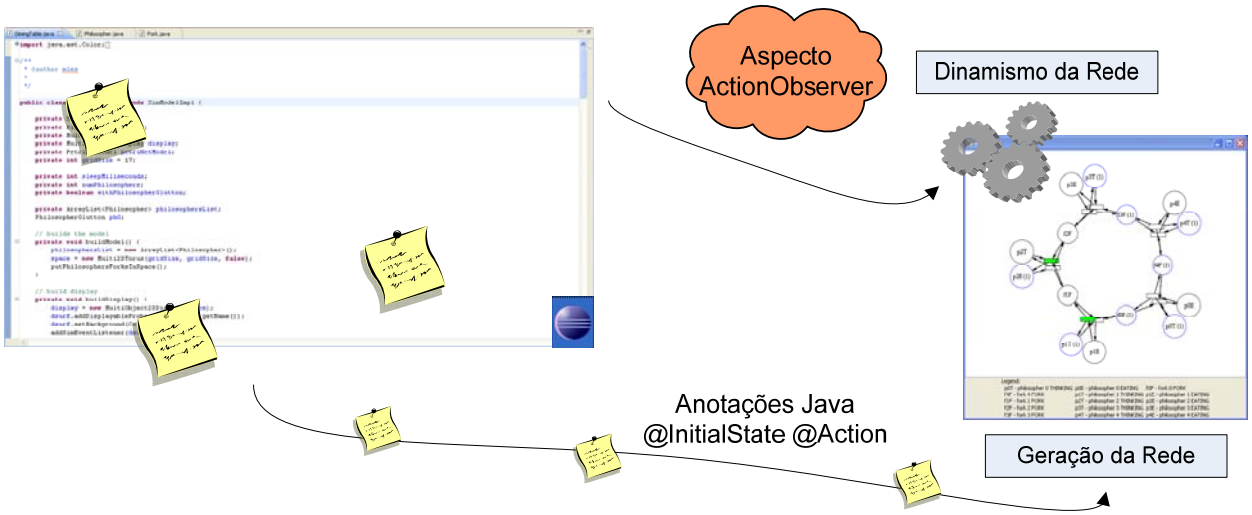


Figura 35. Interface Aplicacional de Programação do módulo Redes de Petri

3.1.3. Apresentação Gráfica

Após o preenchimento do modelo da Rede de Petri (`PetriNetModel`) através da informação fornecida na API pelos mecanismos explicados na secção anterior, foi desenvolvido a apresentação gráfica deste modelo na aplicação *Repast*.

É dado a conhecer, na Figura 36, a aparência final da representação gráfica do exemplo utilizado, o Jantar de Filósofos. É fácil identificar as duas janelas de exposição gráfica da biblioteca: a janela de mundo da simulação onde se encontra a mesa de jantar dos filósofos (à direita) e a janela da Rede de Petri sincronizada (à esquerda). Para realizar a aparência gráfica da rede foi utilizado um software externo denominado *Graphviz* [Ellson et al., 2003].

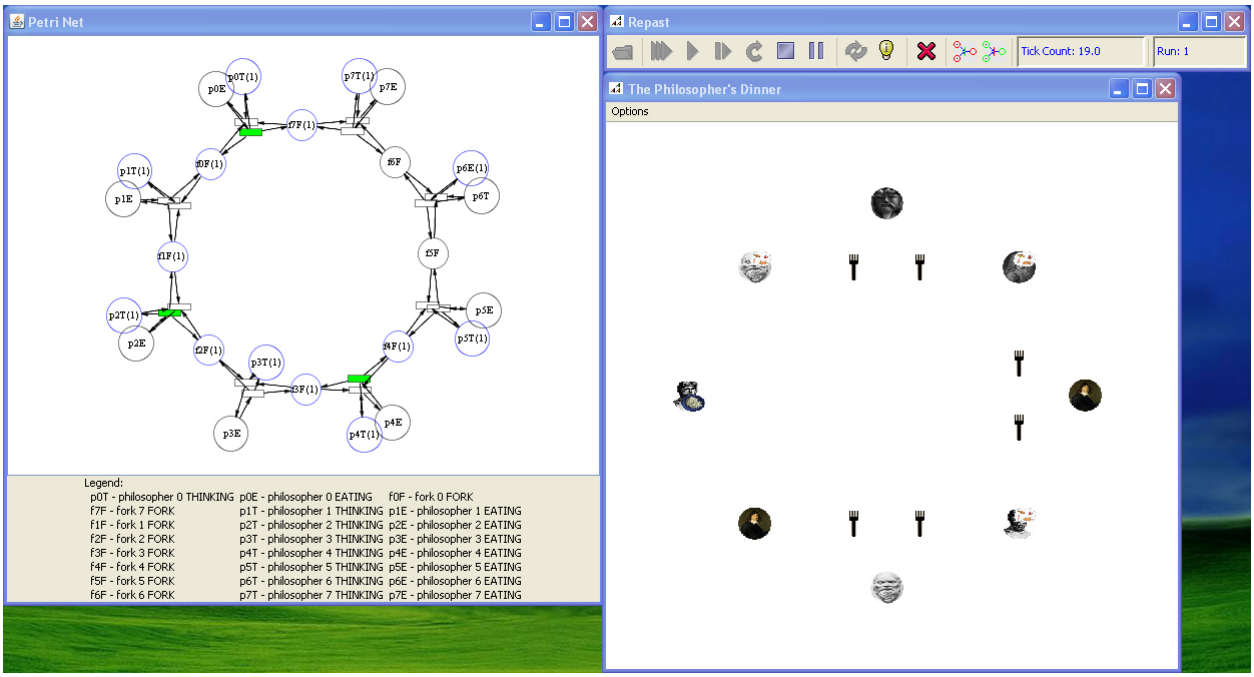


Figura 36. Apresentação gráfica da simulação

Graphviz é uma aplicação de visualização de grafos de utilização livre e o seu código fonte é fornecido gratuitamente. A sua visualização é uma forma de representação estrutural de informação em diagramas de redes e grafos abstractos. Existe um pacote específico Java que simplifica a sua integração com aplicações Java, que contém, ainda, a possibilidade da extensão das suas funcionalidades, visto que o seu código fonte é fornecido. Esta plataforma abrange todo um conjunto de funções de construção e manipulação de grafos, sub-grafos e respectivos nós com atributos próprios. Existem vários algoritmos que implementam diferentes distribuições gráficas dos nós das redes a representar.

Com a utilização da ferramenta *Graphviz*, foi dada ao utilizador *Repast* a possibilidade da visualização de três Redes de Petri diferentes resultantes da simulação:

- Rede de Petri em tempo real
- Rede de Petri de frequência de lugares marcados
- Rede de Petri de frequência de disparo de transições

3.1.3.1. Rede de Petri em tempo real

A primeira Rede de Petri corresponde ao modelo do problema em simulação. A Figura 37 apresenta o resultado do exemplo utilizado nesta dissertação. É a rede consequente do contexto modelado. É animada em tempo real e sincronizada com o decorrer da simulação.

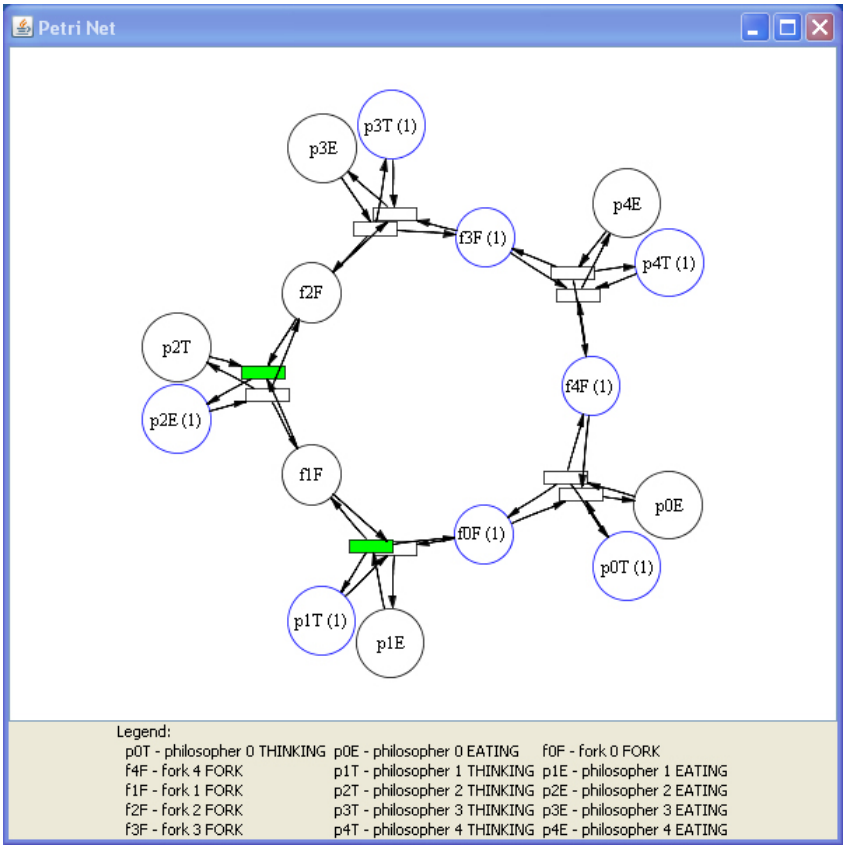


Figura 37. Rede de Petri em tempo real

No exemplo da Figura 37, estão visíveis todos os nós, estando os lugares representados por círculos ocos, as transições afiguradas por barras (rectângulos) e os respectivos arcos representados por ligações directas entre lugares e transições. As marcas dos lugares, em vez de serem círculos pequenos e sólidos, estão indicados textualmente entre parêntesis, no interior de cada lugar. Para aumentar a percepção e simular algum dinamismo, as transições recentemente disparadas apresentam uma cor verde. Os lugares marcados, isto é, com um número de marcas superior a zero, ostentam uma cor azul como é possível verificar na Figura.

Na rede ilustrada, do jantar de cinco filósofos, é identificado, para cada um deles, dois lugares para os estados “a comer” e “a pensar” e outros dois lugares para os garfos esquerdo e direito. São visíveis também as respectivas transições entre esses mesmos lugares e os arcos que

os interligam. Na parte inferior da Rede de Petri existe uma legenda que descreve os nós existentes.

A nomenclatura dos nós da rede seguiu uma sintaxe própria que será mais fácil de compreender com um exemplo concreto: Um filósofo, instância da classe `Philosopher` que, resultante da implementação da interface `PetriElement`, contém os elementos `identifier` e `description` com valores *"p1"* e *"philosopher 1"* respectivamente. Neste mesmo filósofo existem referências para outras duas instâncias da classe `Fork` que representam os seus garfos esquerdo e direito na mesa. Estas instâncias de garfo contêm também os elementos `identifier` e `description` com valores "f0", "f1" e "fork 0", "fork 1" respectivamente. Na classe `Philosopher` existem anotações que definem os estados `"THINKING"` e `"EATING"` e a classe `Fork` contém a anotação que define o estado `"FORK"`. Com esta configuração, para este exemplo concreto, resultam os seguintes lugares e legenda:

- Lugar *"p1T"*; Legenda *"philosopher 1 THINKING"* – Estado do filósofo a pensar. O nome do nó provém da concatenação do elemento `identifier` ("p1") do objecto com a letra inicial do seu estado `"THINKING"`. A descrição, que se encontra em Legenda, deriva da concatenação entre o elemento `description` ("philosopher 1") e o nome do seu estado `"THINKING"`.
- Lugar *"p1E"*; Legenda *"philosopher 1 EATING"* – Estado do filósofo a comer. Situação idêntica à anterior, onde o estado é `"EATING"`.
- Lugar *"f0F" "f1F"*; Legenda *"fork 0 FORK" "fork 1 FORK"* – Estado dos garfos na mesa. Novamente idêntica à anterior, onde os objectos em causa são da classe `Fork`, sendo utilizado os respectivos atributos de identificação e descrição.

Com esta Rede de Petri, os possíveis investigadores têm um novo resultado gráfico dos seus problemas simulados, sendo este grafismo actualizado em tempo real.

3.1.3.2. Redes de Petri de Frequência

Com base na Rede de Petri em tempo real, foram criadas outras duas redes:

1. A Rede de Petri de frequência de lugares marcados.
2. A Rede de Petri de frequência de disparo de transições.

Estes modelos têm o objectivo principal de fornecer uma nova ferramenta gráfico-analítica do caminho seguido por uma simulação. Por outras palavras, com estas redes de

frequência, o investigador tem uma ilustração de quais os estados ou acções e com que frequência são evidenciados pelas entidades envolvidas na simulação.

No exemplo do Jantar de Filósofos, foram adicionados à barra de ferramentas *Repast* dois botões de acesso às redes de frequência (visível na Figura 38):

- Botão de acesso à frequência de lugares, à esquerda e a vermelho ( )
- Botão de acesso à frequência de transições, à direita e a verde ( ).

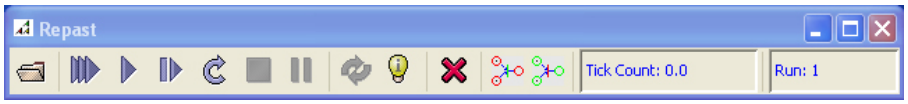


Figura 38. Botões adicionais na barra de ferramentas Repast

A Rede de Petri de frequência de lugares marcados, em termos de estrutura, é idêntica à rede simples em tempo real. Têm os mesmos nós e arcos entre eles. Como se vê na Figura 39, a diferença gráfica reside na cor dos lugares e na sua marcação.

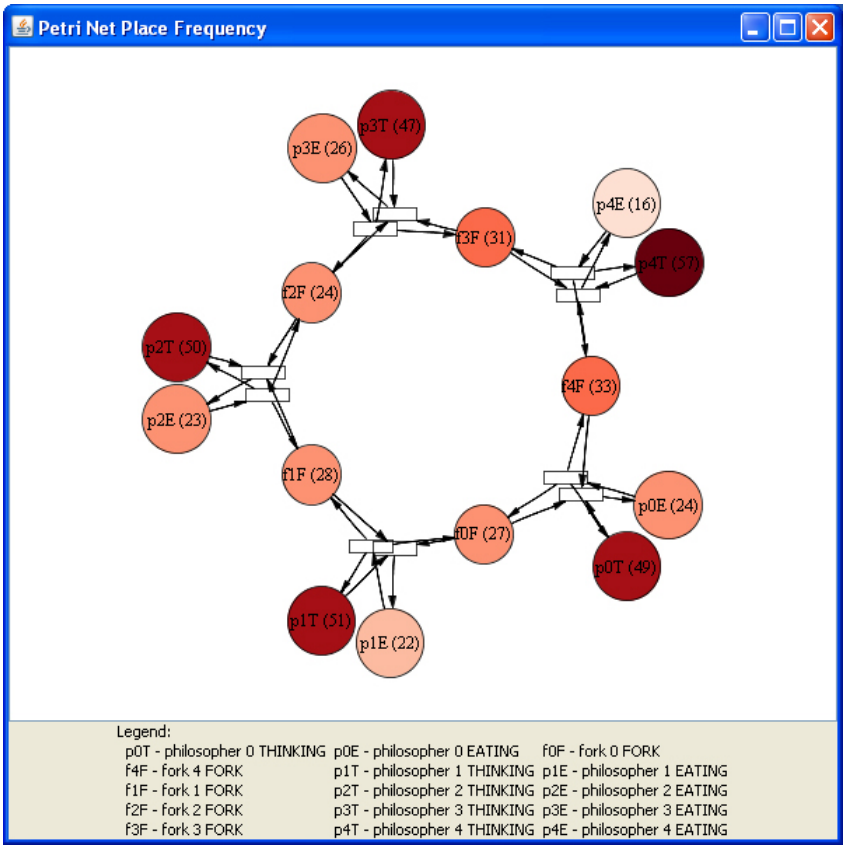


Figura 39. Rede de Petri de frequência de lugares marcados

Para cada lugar, a quantidade que se encontra entre parêntesis é representativa do número de vezes em que a entidade em questão se encontrou naquele estado, durante a simulação. Por exemplo, através da Figura 39, é verificado que o filósofo "3" esteve a "pensar" durante 47 momentos distintos durante a simulação. Isto é, a marcação desse lugar identificado por "p3T" esteve positiva em 47 tiquetaques de relógio da simulação. Ou seja, o lugar esteve marcado em 47 instantes. O tiquetaque de relógio é a unidade temporal da simulação *Repast*.

Para salientar a diferença quantitativa de frequência de marcações positivas na rede, foram utilizadas cores distintas. A Figura 40 ilustra o espectro de cores avermelhadas utilizado. A cor mais esbatida corresponde à menor frequência e a cor mais carregada corresponde à maior frequência evidenciada em toda a rede.

Figura 40. Esquema de cores

A segunda rede de frequências, a Rede de Petri de frequência de disparo de transições, em termos de estrutura, é novamente idêntica à anterior. A diferença gráfica não reside no esquema de cores, mas sim em que nós esse espectro de cor é patenteado: nas transições da rede.

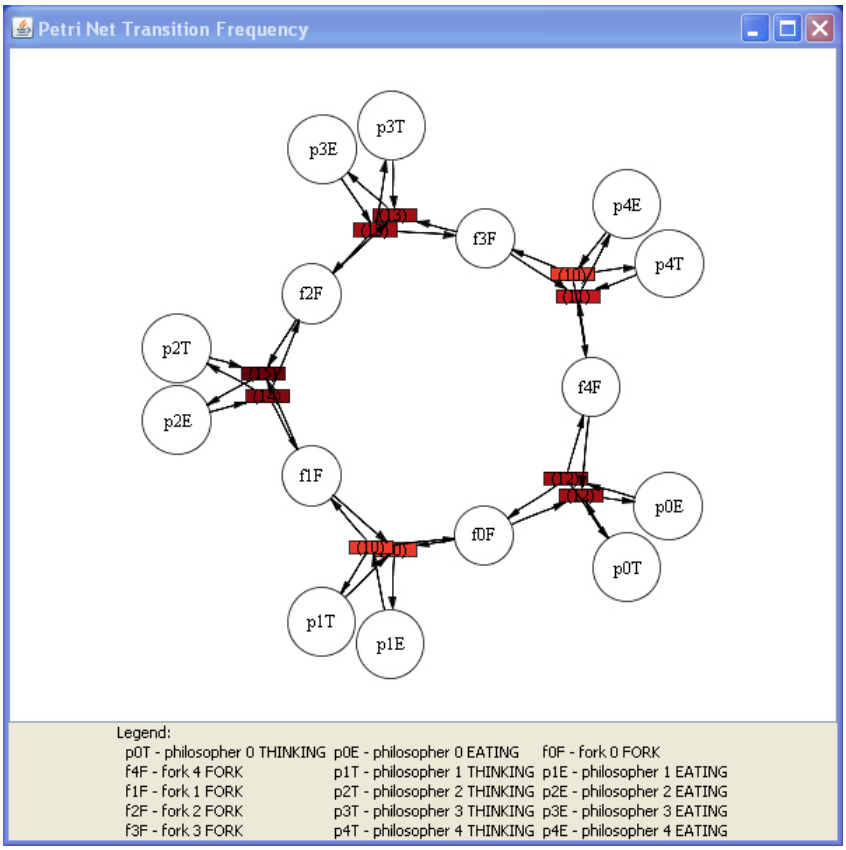


Figura 41. Rede de Petri de frequência de disparo de transições

Como está observável na Figura 41, para cada transição, o número que é escrito entre parêntesis expressa a quantidade de vezes em que a entidade executou a referida acção (ou transição de estado) durante a simulação. Por exemplo, o filósofo "4" parou de comer e iniciou o seu pensamento em 10 instantes distintos. Isto é, a transição correspondente foi disparada 10 vezes na Rede de Petri da simulação.

Depois de desenvolvida a proposta, isto é, depois da implementação das Redes de Petri como consequência de simulações multi-agente, foi necessário examiná-las. Foram validadas e testadas, para provar que têm qualidade. Estas simulações e testes estão relatados no capítulo seguinte.

4. VALIDAÇÃO

Neste Capítulo, são apresentados os testes, simulações e respectivas conclusões realizadas sobre as Redes de Petri em tempo real e sobre as Redes de frequência.

A secção 4.1 valida a Rede de Petri em tempo real, enquanto que 4.2 confirma a validade das Redes de Petri de frequência. Finalmente, na secção 4.3 deste capítulo, é apresentado um modelo ligeiramente diferente do original problema do Jantar de Filósofos, realizado para a detecção de ocorrência de impasses na Rede de Petri.

Neste novo modelo alterado, foi acrescentado um novo filósofo central denominado glutão. O principal interesse da apresentação deste modelo reside na possibilidade de, simultaneamente, apresentar a funcionalidade implementada para a detecção de bloqueios na Rede de Petri. Foi necessário criar o modelo do filósofo glutão porque na Rede de Petri a representar o modelo original do Jantar, nunca existem marcações que impossibilitem o disparo de transições (nunca ocorrem impasses).

4.1. Rede de Petri em tempo real

Para validar a Rede de Petri em tempo real, resultante directamente do modelo simulado, foram testadas simulações do Jantar de Filósofos com diferente número de agentes existentes a representar filósofos à volta da mesa de jantar. O objectivo foi perceber se a rede representava na perfeição a situação real do modelo em causa. O Anexo C apresenta as tabelas C.1, C.2, C.3 e C.4 que contêm os resultados utilizados para efectuar a validação das Redes de Petri em tempo real.

Pelos resultados obtidos, foi concluído que os estados e as acções da simulação eram evidenciados pela rede em tempo real. Para chegar a esta conclusão foram executadas 50 simulações do Jantar com 2, 5 e 10 filósofos.

Nesta secção, são apresentadas algumas imagens retiradas das simulações efectuadas. A Figura 42 mostra a representação de um exemplo de simulação do Jantar para 2 filósofos e a respectiva Rede de Petri em tempo real.

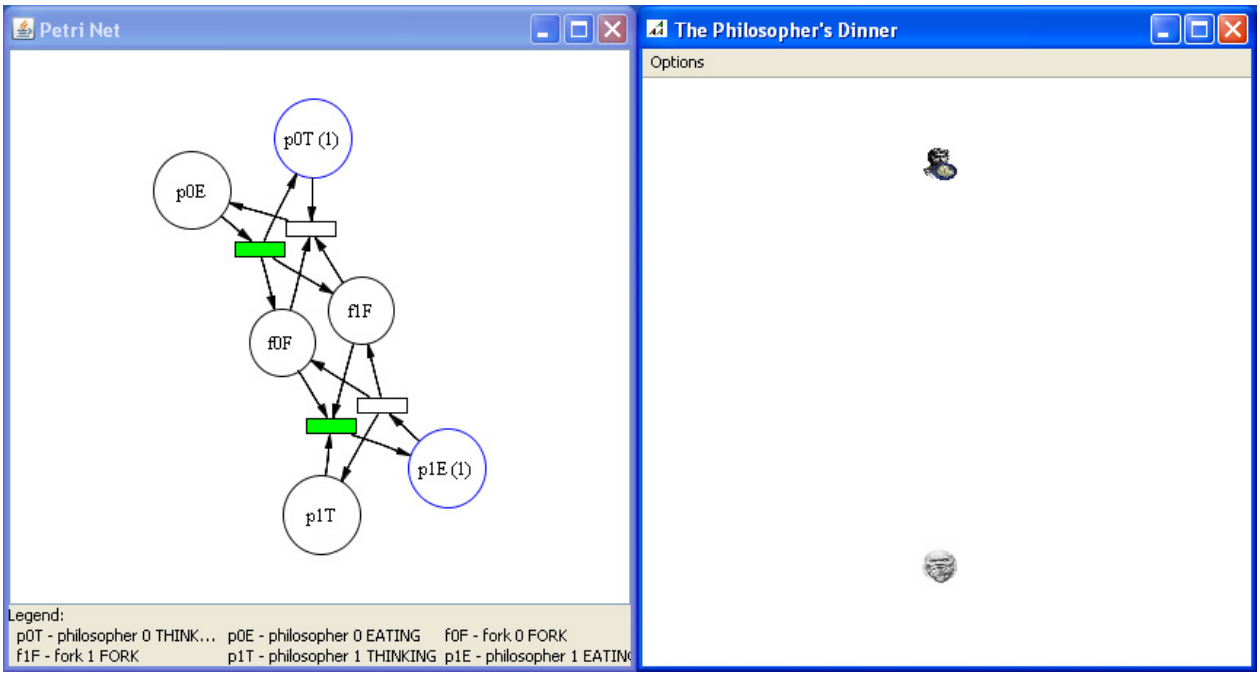


Figura 42. Rede de Petri de dois filósofos

É observável, tanto na rede como no mundo ilustrado que existem apenas dois filósofos, onde um deles se encontra a comer e o outro a pensar. Não existe qualquer garfo na mesa pois foram recolhidos na última acção realizada pelo agente "p1" (começou a comer). Na Rede de Petri, esta última acção realizada tem correspondência na transição que tem como lugar de saída o "p1E" e lugares de entrada "f0F" e "f1F". Esta transição ostenta a cor verde como resultado de ter sido disparada.

A Figura 43 ilustra o estado de uma das 50 simulações do modelo do Jantar parametrizado com 10 filósofos.

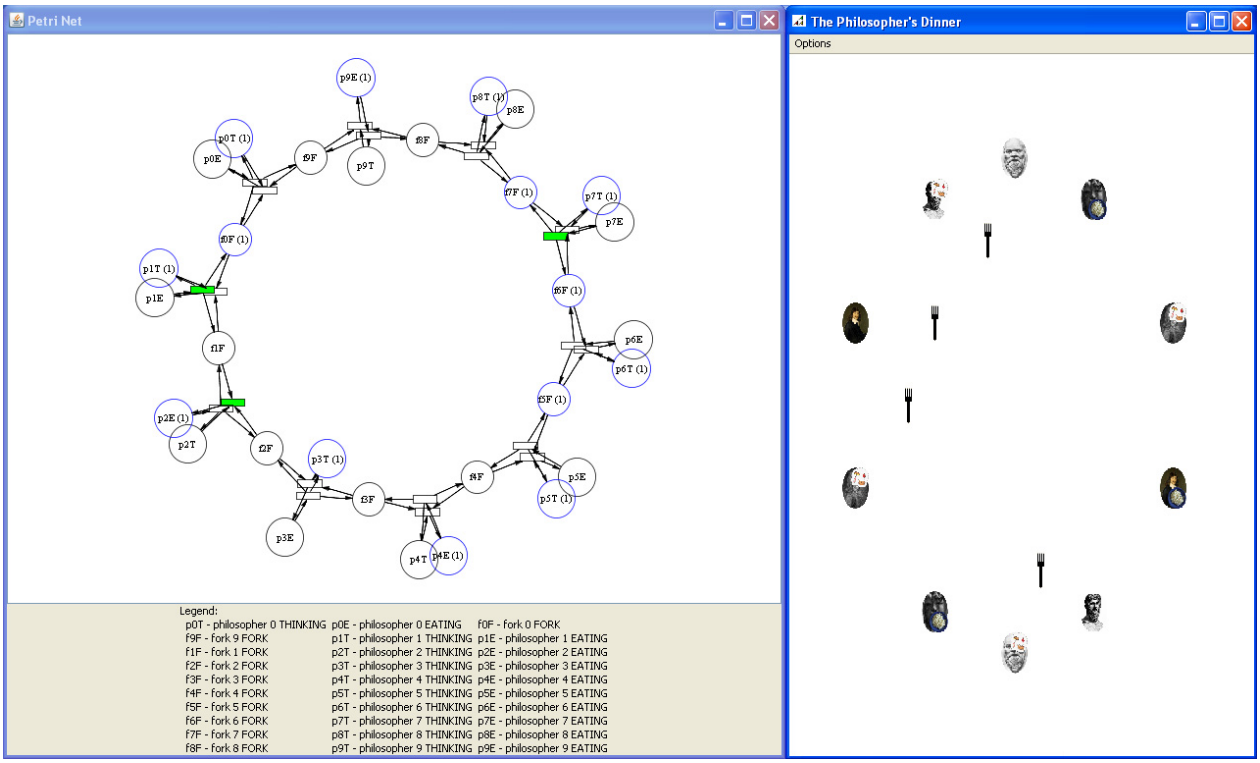


Figura 43. Rede de Petri de dez filósofos

Novamente é perceptível uma perfeita sincronização entre o mundo representado à direita e a respectiva Rede de Petri à esquerda. É fácil presenciar esta correspondência pelos garfos existentes na mesa, onde estão visíveis no mundo representado e, na rede, aparecem representados como quatro lugares marcados de cor azulada (f0F, f5F, f6F, f7F).

Estas duas figuras ilustram apenas dois momentos de dois exemplos dos muitos protótipos, registados no Anexo C., efectuados com o objectivo de validar a estrutura da Rede de Petri, bem como a sincronização desta com o evoluir do estado no modelo simulado. Na validação, ficou concluído que existiu sempre uma fiel representação do modelo, qualquer que fosse o número de agentes e que, a marcação dos lugares e respectiva coloração azul esteve sempre correcta e sincronizada, bem como o disparo de transições e respectiva coloração verde.

4.2. Redes de Petri de frequência

Para confirmar a validade das Redes de Petri de frequência, tanto as de lugares marcados como as de disparo de transições, foram experimentadas simulações com diferente número de

agentes. Foram executadas 50 simulações do Jantar com 3, 4 e 5 filósofos. Para cada simulação, foram registados os resultados ao fim de 250 tiquetaques de relógio (Anexo C).

Depois de realizados os testes de validação da Rede de Petri de frequência de lugares marcados, ficou concluído que o desenrolar e desfechos finais das simulações eram idênticos e não dependiam do número de filósofos existentes na mesa. Em termos de frequência de lugares, foi verificado que a partir de um determinado momento da simulação (ao fim de 250 tiquetaques de relógio) existia uma estabilização da frequência relativa entre a marcação dos vários lugares da rede. Isto é, os lugares representativos do mesmo estado dos diferentes filósofos apresentaram uma marcação semelhante e, logo, uma coloração igualmente semelhante. A Figura 44 retrata esta situação, onde é possível observar uma fotografia da Rede de Petri de frequência de lugares marcados, numa simulação de cinco filósofos ao fim de 250 tiquetaques de relógio.

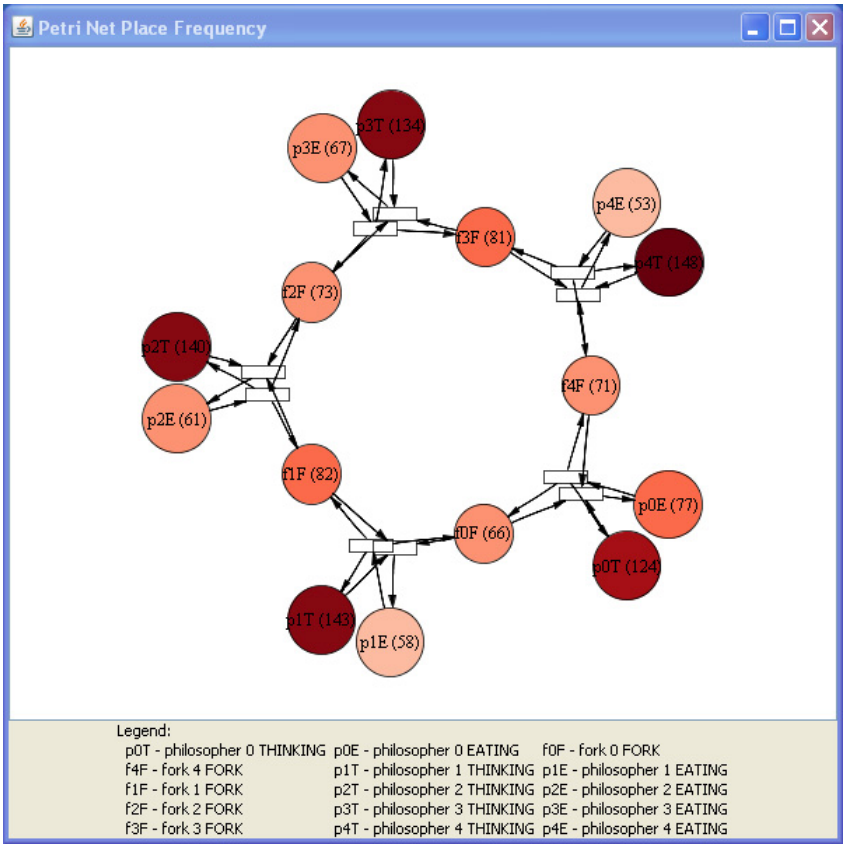


Figura 44. Rede de Petri de frequência de lugares marcados – 250 tiquetaques de relógio

A validação teve como objectivo assegurar dois pontos importantes: verificar se existia uma perfeita correspondência entre a marcação nos lugares da Rede de Petri em tempo real, o contador de tiquetaques do relógio da simulação e os contadores existentes em cada lugar na

Rede de Petri de frequência de lugares marcados. Era conjuntamente necessário averiguar se a coloração da rede de frequência estava correcta com a marcação de cada lugar. Os dois pontos foram testados e comprovados nas simulações efectuada, sendo que os resultados estão registados na tabelas C.5, C.6 e C.7 do Anexo C. Os resultados mostram que, para cada filosofo, a soma dos contadores dos seus lugares na rede é igual ao número de tiquetaques de relógio da simulação (250), o que ajudou a comprovar a validade da rede. Na Figura 44, é possível verificar que a coloração dos nós da rede de frequência se encontra correcta, onde a cor mais esbatida correspondente à menor frequência e a cor mais carregada correspondente à maior frequência evidenciada em toda a rede.

Das análises realizadas, aos resultados obtidos nas redes de frequência de marcação de lugares, foi possível concluir os seguintes aspectos:

- Os filósofos passam mais tempo a pensar do que a comer. A própria coloração da Rede de Petri da Figura 44 transmite esta ideia. O filósofo passa sensivelmente o dobro do tempo a pensar, o que era esperado, visto existir apenas um garfo para cada filósofo, precisando este de dois para comer. O filósofo evidencia o estado "a pensar" em dois terços do tempo da simulação.
- Os garfos estão livres na mesa em menos de metade do tempo da simulação. O tempo de utilização do garfo tende a aumentar ligeiramente com o aumento do número de filósofos à volta da mesa.
- Comparando as várias entidades da simulação entre si, é possível notar que o tempo usado para comer ou pensar é semelhante nos vários filósofos e os tempos dos garfos na mesa são igualmente semelhantes. Por outras palavras, não existe nenhum filósofo que se destaque em relação aos outros no tempo que utiliza para comer ou pensar e, similarmente, não existe nenhum garfo que seja mais ou menos utilizado que os restantes. Isto era espectável, uma vez que, na simulação, não existem características que privilegiem um agente em relação aos outros.

Na validação de Redes de Petri de frequência de disparo de transições, foram também realizadas simulações com diferentes parametrizações, sendo que os resultados estão registados na tabelas C.8, C.9 e C.10 do Anexo C. A Figura 45 mostra o momento final, o tiquetaque de relógio nº 250, de uma das 50 simulações efectuadas com cinco filósofos à volta da mesa.

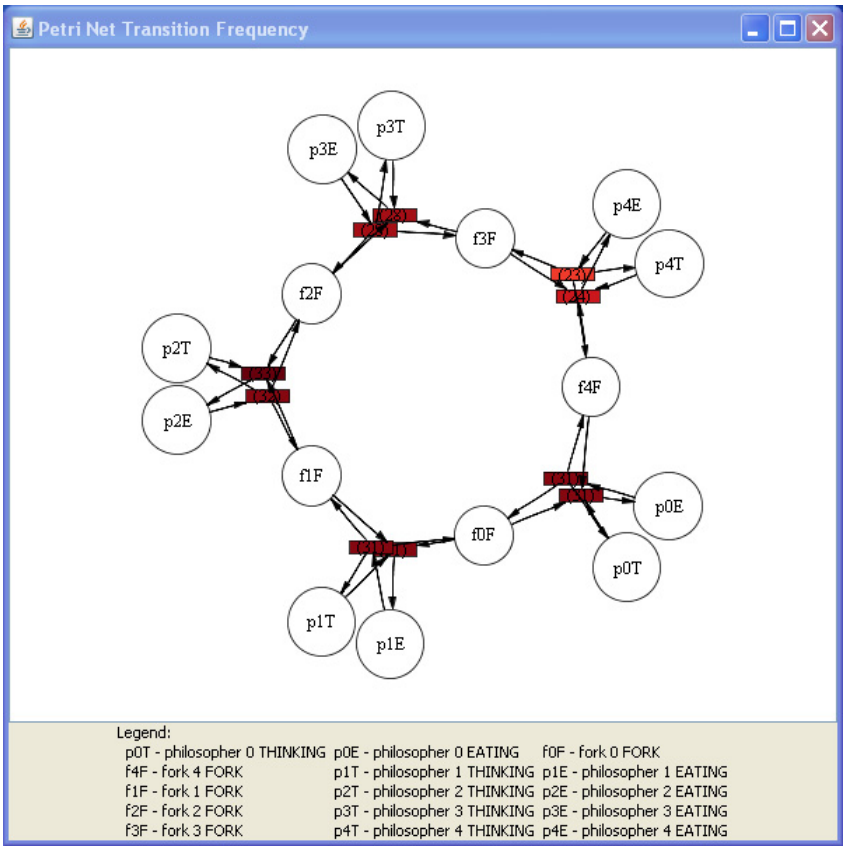


Figura 45. Rede de Petri de frequência de disparo de transições – 250 tiquetaques

Com base nos resultados obtidos foi deduzido que, em termos de frequência de disparos de transições, a partir do primeiro momento (primeiro tiquetaque de relógio) existia uma estabilização da frequência relativa de ocorrência de acções (transições) entre as várias entidades envolvidas na simulação. Ficou concluído, também, que o desenrolar e desfechos finais das simulações eram os mesmos, independentemente do número de filósofos existentes na mesa.

À semelhança da validação da Rede de Petri de frequência de lugares marcados, a validação desta rede teve como objectivo confirmar duas questões importantes: verificar se a coloração na rede correspondia ao número de disparos efectuados pelas transições; e confirmar a existência da relação entre o disparo de cada transição na Rede de Petri em tempo real e o contador existente nessa mesma transição na respectiva Rede de Petri de frequência. Os testes realizados, com distintos parâmetros de simulação, vieram comprovar estas duas questões.

Das simulações executadas e das análises dos resultados das redes de frequência de disparo de transições, foi possível chegar às seguintes conclusões:

- A quantidade de vezes que um filósofo reinicia o seu jantar ou reinicia o seu pensamento é idêntica desde o início da simulação. Por outras palavras, o agente filósofo altera o seu estado para "a comer" o mesmo número de vezes que altera o seu estado para "a pensar". Isto era previsto, uma vez que cada agente apenas apresenta dois estados distintos e vai alternando entre eles.
- Os resultados mostraram que, para cada filósofo, a soma dos contadores das suas transições na rede não é igual ao número de tiquetaques de relógio da simulação (250). Esta desigualdade não invalida a rede, isto porque, em determinados momentos da simulação, o filosofo não executa qualquer tipo de acção.
- Confrontando os vários agentes entre si, é observável que o número de execuções das acções é semelhante a todos os filósofos. Isto é, não existe nenhum filósofo que se destaque em relação aos restantes na quantia de acções que realiza (ir pensar ou ir comer). Isto foi verificado já num contexto avançado da simulação (ao fim de 250 tiquetaques de relógio). Mais uma vez este ponto era espectável uma vez que, na simulação, não existem excepções que privilegiem um agente em relação aos outros.

4.3. Detecção de bloqueios

Nesta secção, é introduzida uma nova funcionalidade implementada na Rede de Petri em tempo real. A rede tem a capacidade de, em cada tiquetaque da simulação, testar a sua vivacidade. A vivacidade de uma rede é a existência ou não da possibilidade de disparo das transições que a compõem. Em todos os tiquetaques de relógio da simulação, é verificado se todas as transições existentes na rede se encontram mortas. Uma transição está morta caso nunca esteja habilitada, qualquer que seja a marcação da Rede (a transição não poderá mais ser disparada).

A propriedade vivacidade adiciona a capacidade de detecção de impasses na aplicação, podendo-se, assim, observar se a simulação chegou a uma situação de bloqueio, onde todos os seus agentes não mais realizarão qualquer acção e se manterão no mesmo estado, desde esse momento até ao final da simulação.

Para que o modelo do Jantar de Filósofos chegasse a um impasse, foi necessário efectuar algumas alterações. Foi criado um exemplo em que, para além de existirem os vários agentes em

volta da mesa, aparece também um agente central com características diferentes. Este filósofo, denominado glutão, necessita de todos os garfos da mesa para iniciar a sua refeição e, assim que a inicia, nunca mais pára de comer, daí o seu nome. No *Repast*, na janela de parametrização, para dar início a uma simulação desta variância, foi acrescentada a opção indicadora da existência de um especial filósofo glutão. A Figura 46 exibe um exemplo da Janela de representação gráfica da simulação, no estado em que o filósofo central se alimenta, onde não existe qualquer garfo na mesa e todos os restantes filósofos se encontram a pensar com fome.

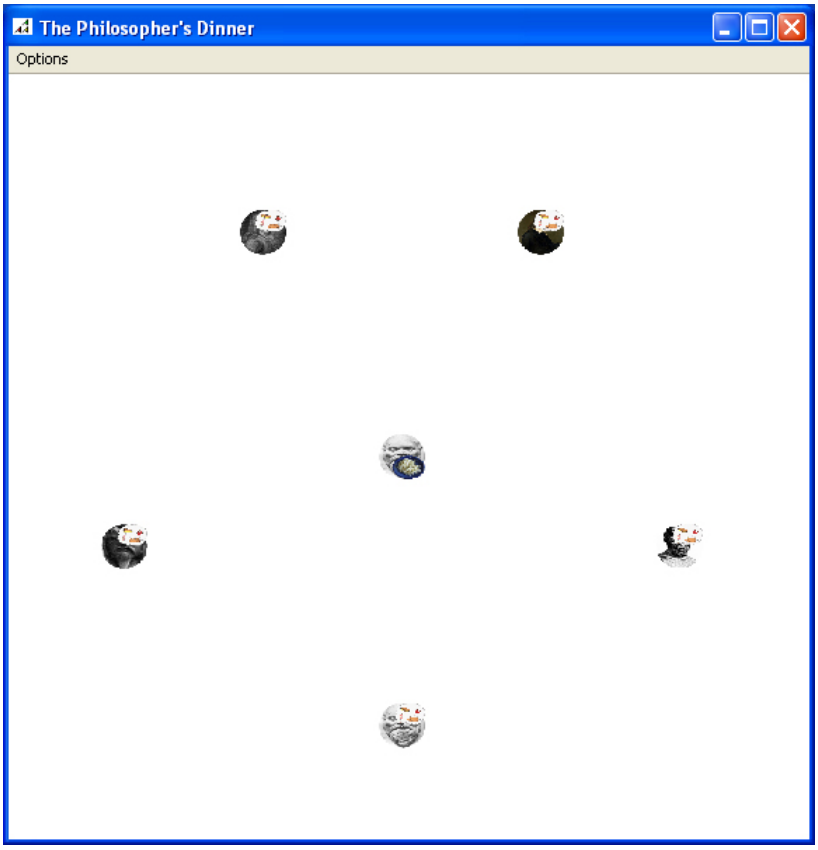


Figura 46. Simulação com filósofo glutão

Em caso de interrupção da rede, é lançada uma mensagem a avisar que existe uma situação de bloqueio. A projecção desta mensagem interrompe a simulação. A Figura 47 ilustra um exemplo de uma rede morta com a respectiva mensagem de aviso.

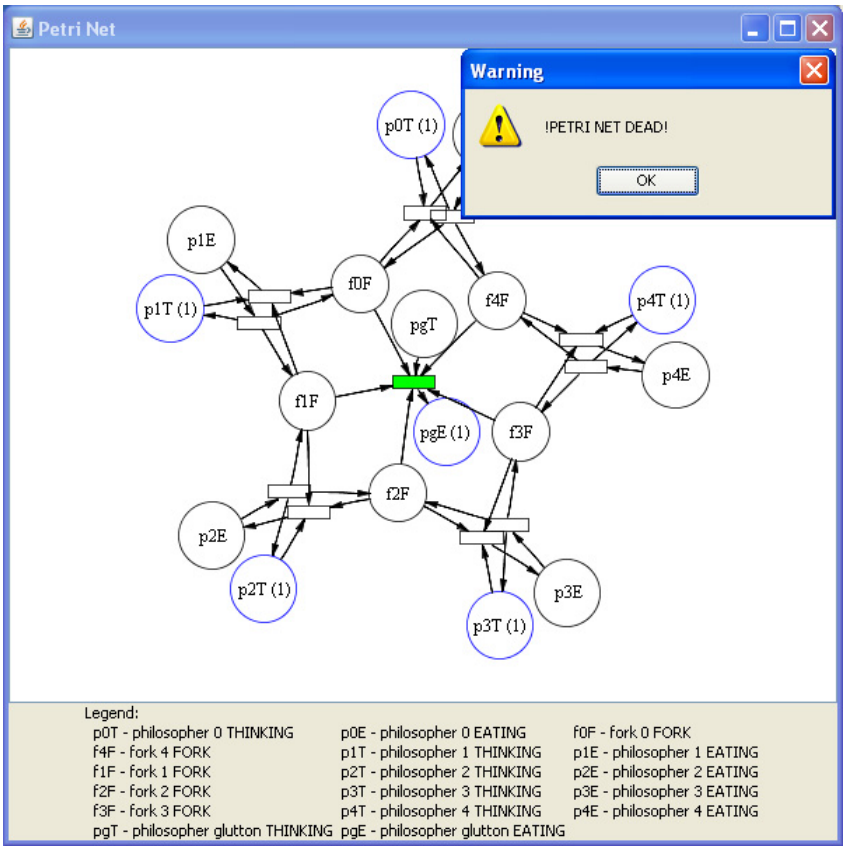


Figura 47. Rede de Petri Morta

Foram testadas 50 simulações desta variância do Jantar de Filósofos, com 2, 5 e 8 filósofos em volta da mesa, em que se registou o número de tiquetaque de relógio quando ocorreu o bloqueio da rede (Tabela C.11 do Anexo C). Em todas as simulações, o impasse acabou sempre por surgir quando o agente glutão iniciou a sua refeição. Com o aumentar da quantia de filósofos existentes no modelo, a ocorrência do impasse tendeu a acontecer mais tardiamente na simulação. Este comportamento é explicado pelo facto de o filósofo glutão necessitar de todos os garfos livres na mesa, sendo que o número de garfos aumenta com o número de filósofos parametrizado na simulação.

Nos testes realizados, quando o agente glutão era o primeiro a obter os garfos disponíveis na mesa, a Rede de Petri morria logo no primeiro tiquetaque de relógio da simulação.

Nas simulações efectuadas sobre este novo modelo do jantar com filósofo glutão, foram igualmente analisadas as Redes de Petri de frequência resultantes. Para isto foram realizadas 50 simulações deste modelo do Jantar com 5 filósofos. Para cada simulação, foi registada a

marcação das redes de frequência, quando ocorreu o impasse (Tabelas C.12 e C.14 do Anexo C) e ao fim de 300 tiquetaques de relógio (Tabelas C.13 e C.15 do Anexo C).

Da análise aos resultados das redes de frequência de disparo de transições, foram tiradas as seguintes conclusões:

- A transição central, que corresponde à acção de iniciar a refeição por parte do filósofo glutão, apenas efectuou um disparo. Este facto ocorreu em todas as simulações realizadas e já era esperado, visto que é o disparar desta transição que bloqueia toda a Rede de Petri. Na Figura 48 é possível observar este estado.
- Depois da ocorrência do impasse referido no ponto anterior, não mais ocorreu nenhum disparo. A rede de frequência de disparo de transições congelou. Desde o momento de impasse até ao final da simulação, a rede manteve-se inalterada, não ocorrendo mais nenhum disparo das transições central e restantes, evidenciando e comprovando assim a total inactividade da rede.

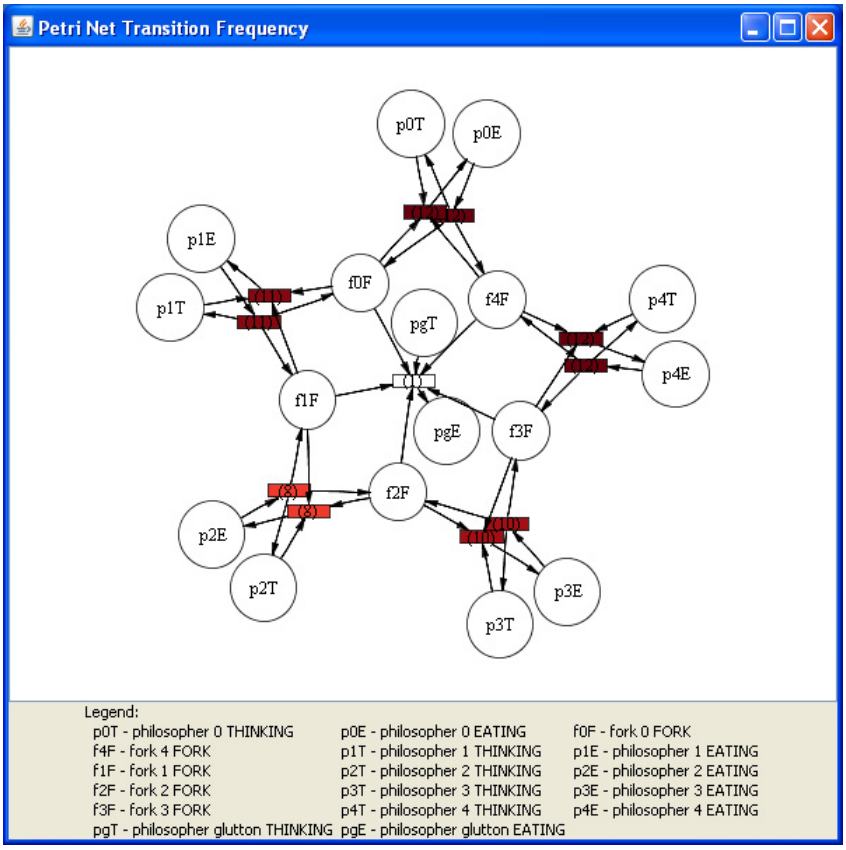


Figura 48. Rede de Petri de frequência de disparo de transições mortas

Da análise dos desfechos das redes de frequência de lugares marcados, foram deduzidas as conclusões que se seguem:

- Antes de ser atingida a situação de bloqueio da rede, a rede de frequência de lugares marcados manifestou as mesmas características do modelo tradicional do Jantar de Filósofos. Nenhum dos filósofos à volta da mesa se destacou em relação aos restantes no tempo que utilizou para comer ou pensar e, analogamente, não existiu nenhum garfo mais utilizado que os outros. Além das características semelhantes ao modelo tradicional, foi verificado que o filosofo glutão central manteve sempre o seu estado "a pensar", algo que era espectável, visto que a situação de impasse ainda não havia sido atingida.
- Assim que foi atingido o bloqueio da simulação, a rede de frequência de lugares marcados alterou completamente o seu comportamento. A rede não congelou mas, deixou de existir alternância de estados (alternância de marcação de lugares). Isto é, os estados evidenciados após o impasse mantiveram-se constantes até ao final da simulação, o que provocou um aumento estável da intensidade de cor vermelha nesses mesmos estados. A Figura 49 ilustra esta situação. À esquerda, é apresentado o estado da rede de frequência, no momento de bloqueio que, neste exemplo, ocorreu ao tique nº 76 de relógio. É observável uma distribuição homogénea do espectro de cores pelos vários lugares da rede. Na figura, à direita, é apresentado o estado da mesma rede de frequência, já no tiquetaque de relógio nº 300 da simulação. Nitidamente, a frequência de marcação é bastante mais intensa em apenas alguns lugares da rede. Estes lugares representam os filósofos em volta da mesa a pensar e o filósofo central a comer, em que esta é precisamente a situação de impasse atingida.

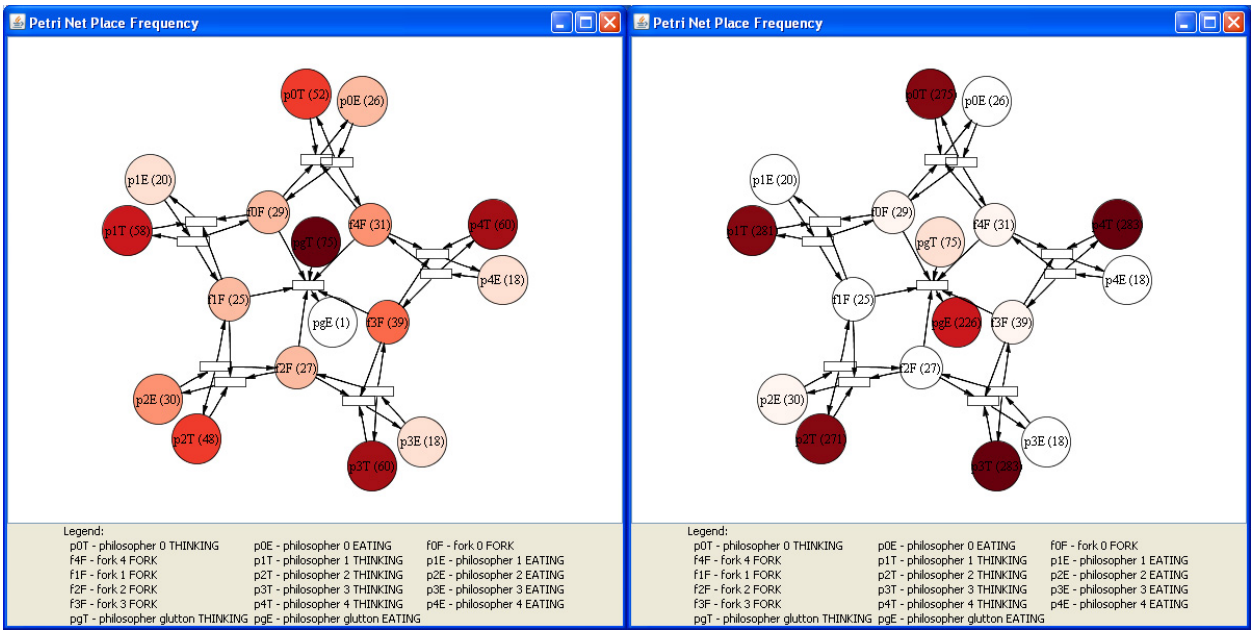


Figura 49. Comparação de Redes de Petri de Frequência de lugares marcados

Este capítulo teve o propósito de validar a qualidade da proposta e, simultaneamente, dar a conhecer algumas das capacidades analíticas do novo módulo de Redes de Petri. Para finalizar a dissertação, é apresentado o capítulo de conclusão, onde se apontam as reflexões finais e são apresentadas ideias para possíveis melhorias e continuação do estudo dos domínios desta dissertação.

5. CONCLUSÃO

Na ferramenta *Repast*, uma das plataformas existentes de simulação multi-agente, além da representação espacial gráfica existente nativa do Repast, foi detectada a necessidade da presença de uma outra representação e modelação gráfica, com noções mais explícitas de concorrência, paralelismo e distribuição de processos.

Esta dissertação propõe as Redes de Petri como ferramenta de representação de simulações multi-agente. Para concretizar esta ideia, foi concebido uma normalização da modelação por Redes de Petri como um módulo de resultados ou consequências de simulações realizadas em *Repast*, que auxilia as investigações simuladas, oferecendo noções gráficas de concorrência e paralelismo.

O modelo de Redes de Petri é uma linguagem de modelação orientada ao processo, adequando-se deste modo uma boa representação gráfica dos resultados obtidos de simulações multi-agente. Como ficou provado, no capítulo de Validação da Proposta, as Redes de Petri auxiliam e enriquecem os testes, validações e análises dos problemas simulados.

As redes auxiliaram a análise do conhecido problema Jantar de Filósofos. Foi utilizado este exemplo, não por ser um problema "real", mas por ser de fácil implementação e pouco complexo, e, simultaneamente, elucidativo de concorrência e sincronização de sistemas multi-agente. O que interessava era o desenvolvimento e validação do novo módulo de Redes de Petri para um exemplo simples e, daí, poder partir para outras simulações de problemas "reais" com mais complexidade.

Foram realizadas simulações sobre um sistema previsível com procedimentos espectáveis. O pretendido não era prever futuros comportamentos ou prognósticos do problema de Jantar de Filósofos. O objectivo era validar a qualidade da nova funcionalidade criada na plataforma *Repast* e, simultaneamente, dar a conhecer possíveis análises que se podem realizar através da modelação por Redes de Petri. O principal interesse da dissertação reside no módulo criado e não no problema simulado. A finalidade é criar a possibilidade de aplicação desta nova ferramenta a qualquer problema simulado, nomeadamente no contexto das ciências sociais.

Esta nova ferramenta gráfica permitiu representar, de uma outra forma, os agentes envolvidos e o respectivo ambiente onde se desenrola a acção. Dá uma outra visão do problema ao investigador, permitindo a este afixar outros resultados das simulações efectuadas.

Assim, como a representação espacial gráfica nativa do *Repast*, que é actualizada em tempo real, com o decorrer de uma determinada simulação, simultaneamente o módulo gráfico de Redes de Petri é actualizado em tempo real e totalmente sincronizado com o estado da simulação em questão. Esta foi uma das características pretendidas e essenciais para o módulo a implementar. Algo que foi conseguido com sucesso e que acrescentou uma noção de dinamismo e sincronismo, sendo uma característica fundamental ao estudo da Rede confrontada com o evoluir da simulação.

A elaboração da dissertação permitiu, ainda, aprofundar conhecimentos na utilização da metodologia de simulação baseada em agentes. Criou, também, a possibilidade de pôr em prática técnicas desenvolvidas recentemente, mais propriamente as novas funcionalidades inerentes da linguagem de programação Java: Anotações Java e Programação Orientada para Aspectos.

A aplicação da programação orientada para aspectos (AOP), combinada com anotações, foi fundamental na realização do novo módulo *Repast*. Isto porque, tecnicamente, era pretendido a injecção de código (de geração de Redes de Petri) em aplicações, ditas externas, criadas por programadores *Repast*. Como resultado da aplicação de AOP, apenas é exigido ao programador (investigador *Repast*) a anotação do seu código fonte, sem a necessidade de alterar a sua lógica, para poder usufruir do novo módulo de Redes de Petri.

A dissertação é inovadora, na medida em que acrescenta a modelação em Redes de Petri a uma plataforma de simulação multi-agente, algo que nunca tinha sido realizado. A forma de disponibilização deste novo módulo ao programador *Repast*, pode igualmente ser considerada inovadora, uma vez que a sua utilização implica um mínimo de alterações ao código fonte da simulação.

Por fim, são apresentadas algumas ideias de possíveis melhorias e de continuação do estudo nos domínios desta dissertação:

- Criar protótipos, testes e simulações de problemas de uma maior complexidade. Isto é, testar e validar o novo módulo desenvolvido com outros modelos a simular que não o já testado Jantar de Filósofos.
- Tentar estender a Rede de Petri clássica, implementada neste trabalho, para Redes Coloridas ou Redes Hierárquicas ou Redes Temporizadas. Estas extensões permitem aumentar o poder computacional das redes tradicionais, possibilitando o acrescento de informação e estruturação ao modelo em rede, algo que simplificaria a modelação de problemas simulados

mais complexos. No capítulo do Estado da Arte, estão documentados exemplos da representação do Jantar de Filósofos em Redes Coloridas e Redes Hierárquicas.

- Implementar perspectivas diferentes de modelação em Redes de Petri. Poderá existir a perspectiva em que a rede representa apenas um agente da simulação, a perspectiva em que a rede representa todo o sistema, a perspectiva em que as marcas da rede representam agentes, entre outras. A ideia é implementar diferentes Redes de Petri, que representem vistas diferentes do mesmo problema e, dar a possibilidade de escolha ao programador cliente.
- Acrescentar ferramentas de análise à Rede de Petri gerada. A detecção de vivacidade da rede (detecção de bloqueios) foi a única ferramenta de análise disponibilizada. Facilmente se poderiam criar outros instrumentos de análise, como, por exemplo, a respectiva árvore ou grafo de cobertura da Rede de Petri. A árvore e/ou grafo de cobertura possibilita a análise de propriedades de uma rede, tais como a Limitação, Segurança, Conservação, entre outras.
- Desenvolver o mesmo módulo de Redes de Petri em tempo real, utilizando outras ferramentas de desenvolvimento de modelos multi-agente que não o *Repast*, e tentar perceber vantagens e desvantagens dessas outras plataformas utilizadas.

Bibliografia

[Axelrod, 2003] AXELROD, R., *Advancing the Art of Simulation in the Social Sciences*, Japanese Journal for Management Information System, 2003.
[Altaweel et al., 2002] ALTAWEEL, M., Blachowicz, D. e Collier, N., *Recursive Porus Agent Simulation Toolkit,* 2002.
[Bakam et al., 2001] BAKAM, I., Kordon, F., Page, C. e Bousquet, F., *Formalization of a Spatialized Multiagent Model Using Coloured Petri Nets for the Study of an Hunting Management System*, FAABS 2000, 2001.
[Barad, 1998] BARAD, M., *Timed Petri Nets as a Verification Tool*, Proceedings of the 1998 Winter Simulation Conference, 1998.
[Barros, 1996] BARROS, J., *CpPNeTS: uma Classe de Redes de Petri de Alto-nível,* 1996.
[Bouyer et al., 2006] BOUYER, P., Haddad, S. e Reynier, P., *Timed Petri Nets and Timed Automata: On the Discriminating Power of Zeno Sequences*, Automata, Languages and Programming, 33rd International Colloquium, 2006.
[Brandão, 2005] BRANDÃO, A., *Um método para estruturação e análise de modelos de sistemas multiagentes baseado em ontologias*, 2005.
[Bressan, 2002] BRESSAN, G., *Modelagem e Simulação de Sistemas Computacionais - Redes de Petri*, 2002.
[Brink, 1996] BRINK, R., *A Petri Net Design, Simulation, and Verification Tool,* 1996.
[Cherkasova et al., 1993] CHERKASOVA, L., Kotov, V. e Rokicki, T., *On Net Modeling of OLTP for Parallel Systems*, 1993.
[Chisty, 2007] CHISTY, M., *An Introduction to Java Annotations,* 2007.
[Cost et al., 1999] COST, R., Chen, Y., Finin, T., Labrou, Y. e Peng, Y., *Modeling Agent Conversations with Colored Petri Nets*, Working notes of the Autonomous Agents '99 Workshop on Specifying and Implementing Conversation Policies, 1999.
[Crespo & Carvalho, 2005] CRESPO, R. e Carvalho, F., *Programação por objectos – UML*, 2005.
[Davidsson, 2002] DAVIDSSON, P., *Agent Based Social Simulation: A Computer Science View*, Journal of Artificial Societies and Social Simulation, 2002.
[Duvigneau et al., 2003] DUVIGNEAU, M., Moldt, D. e Rölke, H., *Concurrent architecture for a multi-agent platform,* Lecture Notes in Computer Science: Agent-Oriented Software Engineering III, 2003.
[Elkoutbi & Keller, 1998] ELKOUTBI, M. e Keller, R., *Modeling Interactive Systems with Hierarchical colored Petri Nets*, Proc. of the Conference on High Performance Computing, 1998.
[Ellson et al., 2003] ELLSON, J., Gansner, E., Koutsofios, E., North, S. e Woodhull, G., *Graphviz and Dynagraph – Static and Dynamic Graph Drawing Tools,* AT&T Labs - Research, 2003.
[Fernandes, 1994] FERNANDES, J., *Redes de Petri e VHDL na Especificação de Controladores Paralelos,* BUM - Dissertações de Mestrado, 1994.
[Fernandes et al., 1997] FERNANDES, J., Adamski, M. e Procnca, A., *VHDL generation from hierarchical Petri net specifications of parallel controllers*, IEE Proceedings, 1997.
[Fernandes & Belo, 1998] FERNANDES, J. e Bello, O., *Modeling Multi-Agent Systems Activities Through Colored Petri Nets An Industrial Production System Case Study*, The 16th International Conference on Applied Informatics, 1998.
[Fernandes & Lima, 2007] FERNANDES, R. e Lima, G., *Hibernate com Anotações,* 2007.

[Filgueira & Costa, 2004] FILGUEIRA, J. e Costa, W., *A Importância de Utilizar UML para Modelar Sistemas: Estudo de caso*, 2004.
[Filman et al., 2004] FILMAN, R., Elrad, T., Clarke, S. e Aksit, M., *Aspect-Oriented Software Development,* Addison-Wesley Professional, 2004.
[Gilbert, 2004] GILBERT, N., *Agent-based social simulation: dealing with complexity*, 2004.
[Gonçalves, 2007] GONÇALVES, R., *Programação Orientada aos Aspectos com AspectJ*, Revista PROGRAMAR, 2007.
[Gradecki & Lesiecki, 2003] GRADECKI, J. e Lesiecki, N., *Mastering AspectJ*, Wiley, 2003.
[Haas, 2004] HAAS, P., *Stochastic Petri Nets for Modelling and Simulation*, Proceedings of the 2004 Winter Simulation Conference, 2004.
[Hertel, 2002] HERTEL, J., *Using Simulations to Promote Learning in Higher Education*, Sterling, Virginia: Stylus, 2002.
[Hoheisel, 2007] HOHEISEL, A., *Grid Workflow - Petri net*, 2007.
[Jamae, 2007] JAMAE, J., *Learn to Use the New Annotation Feature of Java 5.0*, 2007.
[Jayaratchagan, 2004] JAYARATCHAGAN, N., *Declarative Programming in Java*, O'Reilly Network, 2004.
[Jensen, 1998] JENSEN, K., *A Brief Introduction to Coloured Petri Nets*, Workshop on the Applicability of Formal Models, 1998.
[Köhler et al., 2001] KÖHLER, M., Moldt, D. e Rölke, H., *Modeling the structure and behaviour of Petri net agents*, Proceedings of the 22nd International Conference on Application and Theory of Petri Nets, 2001.
[Köhler et al., 2007] KÖHLER, M., Langer, R., Lüde, R., Moldt, D., Rölke, H. e Valk, R., *Socionic Multi-Agent Systems Based on Reflexive Petri Nets and Theories of Social Self-Organisation*, Journal of Artificial Societies and Social Simulation, 2007.
[Kratz, 2000] KRATZ, J., *Unified Modeling Language for Real-Time Systems Design*, 2000.
[Laddad, 2003] LADDAD, R., *AspectJ in Action*, Manning Publications, 2003.
[Lima, 2001]LIMA, *Notas Sobre Redes de Petri*, 2001.
[Lin & Lin, 2006] LIN, F. e Lin, S., *Enhancing the Supply Chain Performance by Integrating Simulated and Physical Agents into Organizational Information Systems*, Journal of Artificial Societies and Social Simulation, 2006.
[Louçã, 2006] LOUÇÃ, J., *Designing agent-based simulations*, Complexity Sciences: Mathematics and Innovation, 2006.
[Louçã, 2007] LOUÇÃ, J., *Modelação UML – Unified Modeling Language*, Mestrado em Ciências da Complexidade, 2007.
[Luke et al., 2004] LUKE, S., Cioffi-Revilla, C., Panait, L. e Sullivan, K., *MASON: A New Multi-Agent Simulation Toolkit*, Proceedings of the 2004 SwarmFest Workshop, 2004.
[Machado et al., 1997] MACHADO, R., Fernandes, J. e Proença, A., *Redes de Petri e VHDL na Prototipagem Rápida de Sistemas Digitais*, 3º Encontro Nacional do Colégio de Engenharia Electrotécnica da Ordem dos Engenheiros, 1997.
[Machado et al., 1998] MACHADO, R., Fernandes, J. e Proenca, A., *Hierarchical Mechanisms for High-level Modeling and Simulation of Digital Systems*, 1998 IEEE International Conference on Electronics, Circuits and Systems, 1998.
[Maciel et al., 1996] MACIEL, P, Lins, R. e Cunha, P., *Introdução às Redes de Petri e Aplicações*, Recife: UFPE, 1996.
[Marranghello, 2005] MARRANGHELLO, N., *Redes de Petri: Conceitos e Aplicações*, Apostila – São Paulo: Universidade Estadual Paulista 2005.

[Marsan et al., 1995] MARSAN, M., Balbo, G., Conte, G., Donatelli, S. e Franceschinis, G., *Modelling with Generalised Stochastic Petri Nets*, John Wiley and Sons, New York, 1995.
[McLaughlin, 2004] MCLAUGHLIN, B., *Add metadata to Java code*, 2004.
[Miczulski, 2001] MICZULSKI, P., *State Space Calculation Algorithm of Hierarchical Petri Nets with Application of Decision Diagrams*, The International Workshop on Discrete-Event System Design, 2001.
[Minar et al., 1996] MINAR, N., Burkhart, R., Langton, C. e Askenazi, M., *The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations*, SFI Working Paper, 1996.
[Miranda & Perkusich, 1999] MIRANDA, M. e Perkusich, A., *Modeling and Analysis of a Multi-Agent System Using Colored Petri Nets*, Proceedings of the Workshop on Application of Petri Nets to Intelligent System Development, 1999.
[Nery et al., 2004] NERY, H., Oliveira, D. e Furtado, V., *AORML uma linguagem para modelagem de uma aplicação Multi-agentes: Uma Aplicação no Sistema Expertcop,* X Encontro de Iniciação à Pesquisa, 2004.
[Penha et al., 2004] PENHA, D., Freitas, H. e Martins, C., *Modelagem de Sistemas Computacionais usando Redes de Petri: aplicação em projeto, análise e avaliação*, Anais da IV Escola Regional de Informática RJ/ES, 2004.
[Peterson, 1981] PETERSON, J., *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, 1981
[Schelling, 1971] SCHELLING, T., *Dynamic Models of Segregation*, Journal of Mathematical Sociology, 1971.
[Silva & Videira, 2005] SILVA, A. e Videira, C., *UML, Metodologias e Ferramentas CASE*, Centro Atlântico, 2005.
[Stewart, 2006] STEWART, D., *Egyptian Hieroglyphics*, 2006.
[Taveter & Wagner, 2001] TAVETER, K. e Wagner, G., *Agent-Oriented Enterprise Modeling Based on Business Rules*, Int. Conf. on Conceptual Modeling, ER2001, 2001.
[Wagner, 2000] WAGNER, G., *Agent-oriented analysis and design of organizational information systems*, Fourth IEEE International Baltic Workshop on Databases and Information Systems, 2000.
[Wagner, 2003] WAGNER, G., *AOR Modelling and Simulation: Towards a General Architecture for Agent-Based Discrete Event Simulation*, AOIS 2003, 2003.
[Weyns & Holvoet, 2002] WEYNS, D. e Holvoet, T., *A Colored Petri Net for a Multi-Agent Application*, Modeling Components, Objects and Agents, MOCA'02, 2002.
[Winck & Goetten, 2007] WINCK, D. e Goetten, V., *AspectJ em 20 minutos*, Novatec Editora, 2007.
[Wooldridge & Jennings, 1998] WOOLDRIDGE, M. e Jennings, N., *Agent Theories, Architectures, and Languages*, Proceedings of ECAI94 Workshop on Agent Theories, 1998.
[Zha et al., 2003] ZHA, X., Lim, S. e Lu, W., *A Knowledge Intensive Multi-Agent System For Cooperative / Collaborative Assembly Modeling and Process Planning*, Journal of Integrated Design and Process Science, 2003.

Anexo A. Anotações Java

Sinteticamente, as anotações (*Java Annotations*) representam informação sobre os dados, isto é, *meta data*. Dão a possibilidade ao programador de criar semântica adicional (dados sobre os próprios dados, informação declarativa) na sua aplicação. Anotam, de uma forma distinta, certos elementos do seu código fonte. Esta funcionalidade foi introduzida na versão 5.0 na linguagem Java. Qualquer símbolo que inicie com uma @ (arroba) é uma anotação, sendo possível associá-las a pacotes, classes, métodos, variáveis, campos e parâmetros [Fernandes & Lima, 2007] [Jayaratchagan, 2004].

Existem dois tipos distintos de anotações: As pré-definidas pela linguagem e as especificadas pelo próprio programador. É apresentada a Tabela A.1, com algumas das anotações pré-definidas existentes. Algumas destas são meta anotações (anotações a ser utilizadas sobre outras anotações) [Jamae, 2007]:

Anotação	Descrição
Deprecated	O compilador alerta para a existência de um elemento no código que está descontinuado
Inherited (meta anotação)	Indica se a própria anotação associada é herdada pelas subclasses
SuppressWarning	O compilador não gera qualquer alerta sobre o código anotado
Retention (meta anotação)	Define a politica de retenção das anotações em memória pela *Virtual Machine Java*
Documented (meta anotação)	Útil para realizar a documentação do código fonte, gerando o respectivo *Javadoc*
Target (meta anotação)	Indica a que elementos se pode associar a própria anotação (variáveis, métodos, entre outros)

Tabela A.1. Anotações pré-definidas

É usada a nomenclatura “anotação” quando o código fonte é anotado (utilização de uma anotação). É usada a nomenclatura “tipo anotação” quando é especificada uma nova anotação (declaração de uma anotação). Esta terminologia é análoga à da classe e respectiva instância de classe. Existem as seguintes regras na especificação de uma anotação:

- Anotações não são extensíveis
- Os seus métodos não podem ter parâmetros
- Não podem ter declarações de lançamento de excepções (*throw exception*)
- Os retornos destes métodos terão de ser de tipos primitivos.

Seguidamente é apresentada uma tabela (Tabela A.2.) com exemplos simples de definições de Anotações (tipos anotações) e respectivas utilizações. Os exemplos Marcador e Elemento único utilizam as meta anotações pré-definidas *Retention* e *Target* [Chisty, 2007] [McLaughlin, 2004].

Categoria	Definição	Utilização

Marcador
(sem elementos)
	`@Retention(RUNTIME)`
	`public @interface Anotacao1 {`
	`}`
		`@ Anotacao1`
		`public void metodo1() {}`

Elemento único
(um elemento)
	`@Target(METHOD)`
	`public @interface Anotacao2 {`
	`  String acto();`
	`}`
		`@Anotacao2 ("acto qualquer")`
		`public void metodo2() {}`

Multi valor
(vários elementos)
	`Public @interface Anotacao3 {`
	`  String acto();`
	`  String data();`
	`}`
		`@Anotacao3 (`
		`acto= "qualquer",`
		`data= "03-11-2007")`
		`public void metodo3() {}`

Tabela A.2. Definições de Anotações

A utilização de anotações cria a possibilidade de processar o código anotado de maneira especial, diferente do restante código, seja pelo compilador, seja em *runtime*, seja por ferramentas de desenvolvimento ou por ferramentas de instalação. Poderá ter as utilizações mais específicas como [Jamae, 2007] [Jayaratchagan, 2004]:

- Gerador de código a partir da meta data.
- Enriquecimento do nível de aviso do compilador.
- Análise de código: utilizar em *runtime* a API Java de *Reflection* (pacote java.lang.reflect) para inspeccionar objectos e respectivas anotações, executando acções específicas (foi realizado nesta dissertação).
- Utilização do paradigma da Programação Orientada para Aspectos, aproveitando secções (*pointcuts*) ligadas a certas anotações declaradas (foi realizado nesta dissertação).

- Gerador de documentação a partir da meta data.
- *Frameworks* de testes unitários utilizando anotações. Exemplo: NUnit.
- Enriquecimento da semântica do código de uma maneira declarativa.

Em suma, existe a possibilidade de desenvolver numa linguagem imperativa juntamente com uma linguagem declarativa e daí tirar os seus benefícios.

Anexo B. Programação orientada para Aspectos

A programação orientada a objectos (POO) tem evoluído muito, mas nem sempre é a melhor aproximação na resolução de todos os problemas. A POO efectua uma separação de conceitos com base no tipo de dados e no padrão de funções que utilizam esses mesmos dados. No entanto, existem preocupações que seccionam a decomposição principal. Por outras palavras, as aplicações contêm questões que lhes são transversais. A implementação de *logging* é um exemplo. Estes "aspectos" transversais da aplicação originam o emaranhamento do código (várias preocupações no mesmo módulo) e a distribuição do código (a mesma preocupação em vários módulos). O exemplo de *logging* tem código fonte associado que geralmente se encontra disperso por todos os módulos, o que complica a administração e a evolução / actualização da aplicação. Este exemplo pode ser visualizado na Figura B.1. São mostradas as linhas de código do componente Tomcat de Apache. Para resolver estas dificuldades surgiu a programação orientada para aspectos (POA) [Gonçalves, 2007] [Gradecki & Lesiecki, 2003].

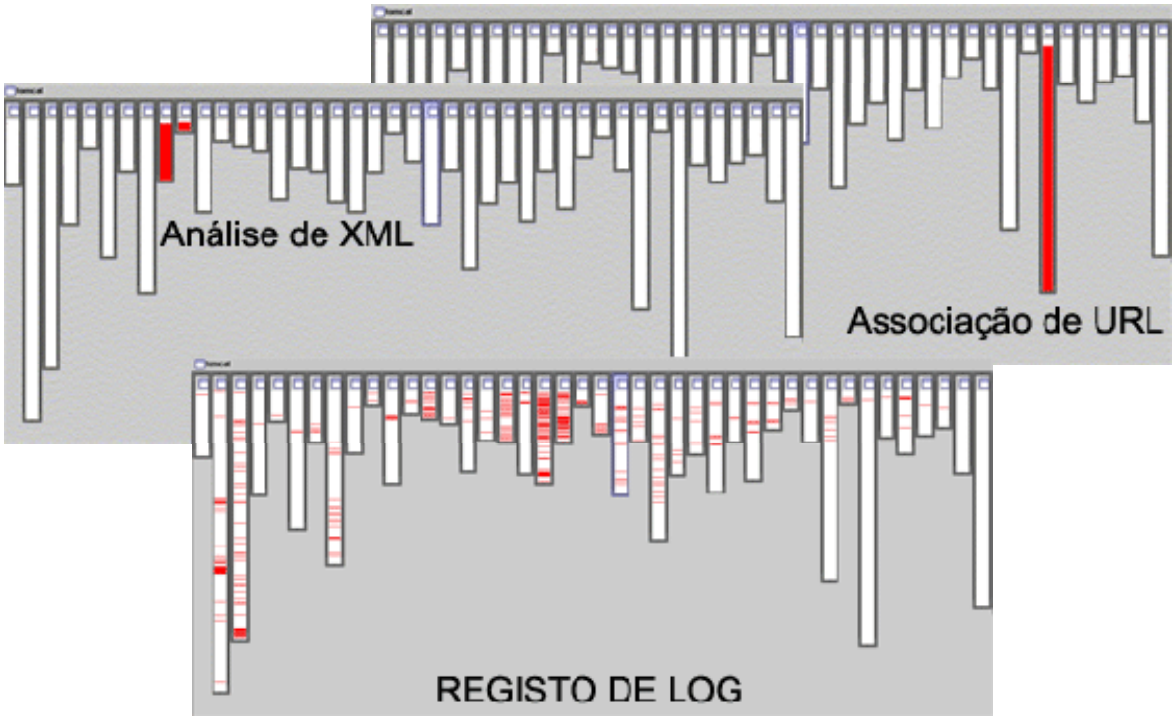


Figura B.1. Linhas de código do componente Apache Tomcat

A programação orientada para aspectos surgiu em 1997 por Greger Kiczalos e os restantes elementos da sua equipa na Xerox. Foram, também, os criadores da linguagem AspectJ (linguagem POA mais utilizada), que permitiu o acrescento de programação de aspectos para aplicações Java. A POA tem como principal propósito separar o código alusivo ao negócio do sistema, dos interesses transversais, de forma delimitada e centralizada. É utilizada uma abordagem que permite à linguagem de desenvolvimento ser composta por uma linguagem principal e várias linguagens específicas. Nestas linguagens especificas são expostas as características sistémicas (ortogonais ou transversais). Permite ao programador a abstracção e o alheamento destes interesses inter-cortantes. O mesmo programador apenas lida com o que é essencial em cada passo e, nos módulos da decomposição principal, não sabe, nem precisa geralmente de saber que código aspectual é entretecido no seu código. A Figura B.2 mostra a separação de conceitos [Winck & Goetten, 2007] [Laddad, 2003].

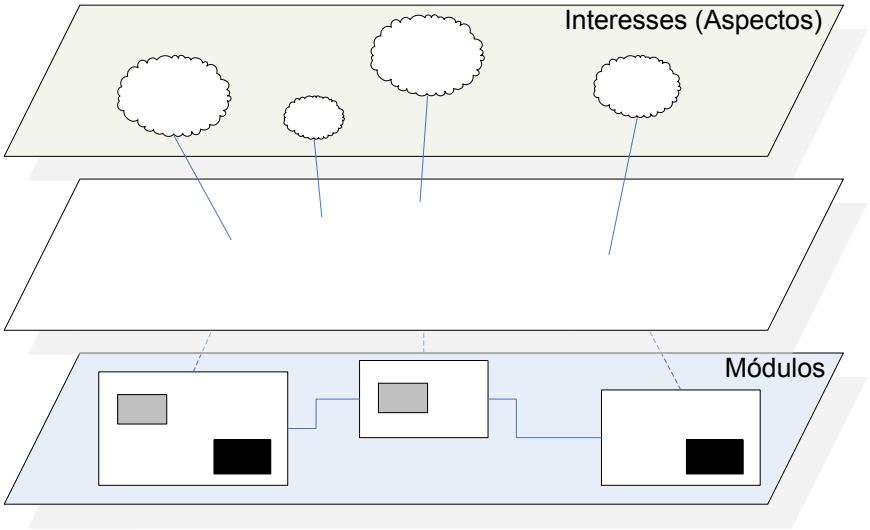


Figura B.2. Separação de conceitos na Programação orientada para Aspectos

A Programação orientada para Aspectos contém três conceitos fundamentais [Winck & Goetten, 2007] [Filman et al., 2004]:

1. Pontos de junção (*Join Points*) – Pontos da aplicação onde os conselhos (*advices*) são realizados: podem ser chamadas a métodos, ocorrências de excepções, acesso a membros, criação de objectos, entre outros.

2. Pontos de actuação / Secção (*Pointcuts*) – Regras genéricas para especificar as ocorrências que serão consideradas pontos de junção da aplicação. Contêm a seguinte sintaxe: *pointcut* <nome>() : <tipo>(<padrão>). Alguns exemplos:
    a. pointcut tostr() : execution(String toString()) – qualquer execução do método toString.
    b. pointcut sets() : call(* Qq*.set*(..)) – qualquer chamada a métodos começados pela palavra set de classes iniciadas pela palavra Qq.
    c. pointcut gets() : get(private String a*) – qualquer acesso de consulta a atributos private do tipo String, iniciados pela letra a.
3. Conselho (*Advise*) – Conselho é a implementação de um aspecto executado em pontos de junção da aplicação principal (*join points*). Tem dois compostos:
    a. O ponto de actuação / secção (*pointcuts*), que descreve as fórmulas de captura dos pontos de junção
    b. O código que será efectuado quando ocorrer o ponto de junção determinado pelo primeiro composto.

A compilação de um programa numa linguagem por aspectos é ligeiramente diferente. Existe um processo, denominado de *weaving* que é responsável por combinar os elementos escritos em linguagem estruturada com os elementos em linguagem de aspectos, nos respectivos pontos de junção. O código do negócio não sofre qualquer modificação para suportar os aspectos A Figura B.3 mostra este processo [Gonçalves, 2007] [Winck & Goetten, 2007].

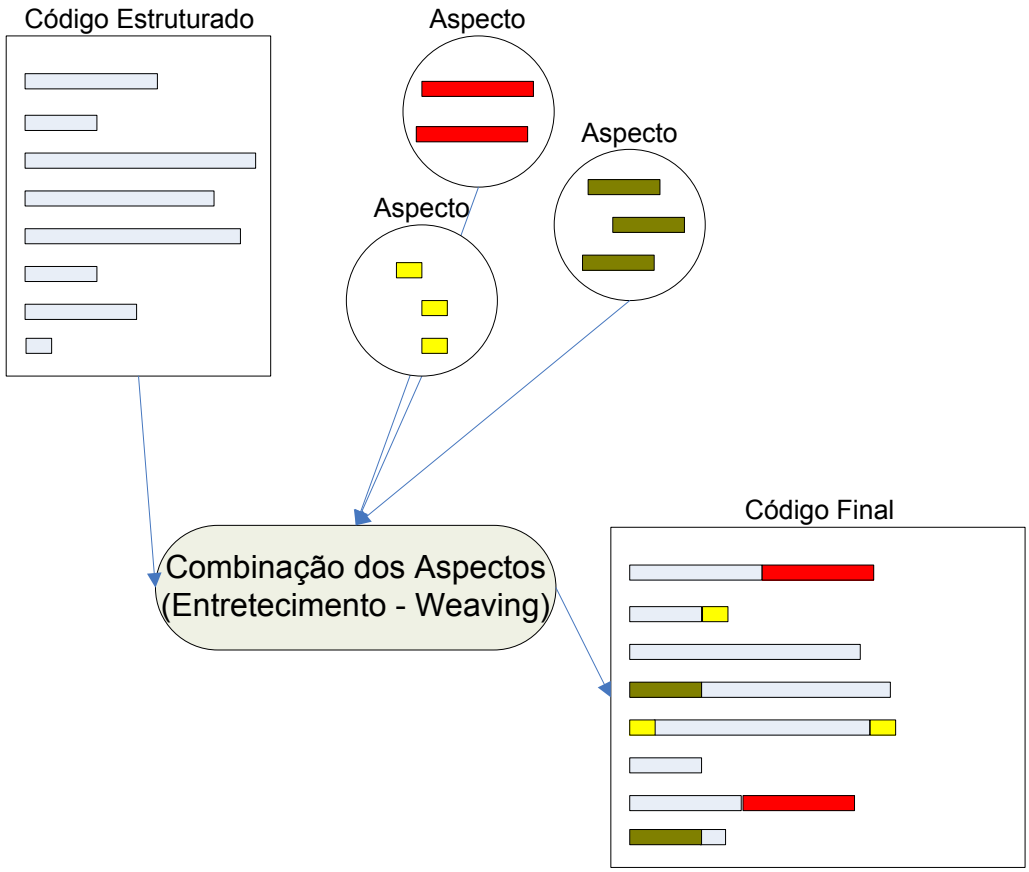


Figura B.3. Compilação na POA

A Programação orientada para Aspectos permite encapsular num único local as características transversais (*crosscutting*), usuais numa aplicação tais como: auditoria (log), tratamento de excepções, persistência, distribuição e sincronização, rastreamento de operações (trace), autenticação e controle de acessos, entre outros.

No contexto da dissertação, o módulo de Redes de Petri é uma característica transversal às aplicações de simulação *Repast*, sendo ideal para a aplicação da tecnologia AOP. É somente necessário ao programador *Repast* efectuar a anotação do seu código fonte, sem a necessidade de modificar a sua lógica.

Anexo C. Tabelas de Resultados de Simulação

Neste anexo, são apresentadas as tabelas que contêm os resultados das simulações efectuadas. Estes resultados foram obtidos na realização da Validação da proposta.

As tabelas C.1 a C.4 exibem resultados que comparam os estados evidenciados pelas entidades da simulação e a respectiva marcação da Rede de Petri em tempo real. Foram executadas simulações do Jantar com 2 filósofos (Tabela C.1), 5 filósofos (Tabela C.2) e 10 filósofos (Tabela C.3). Para cada simulação, foram assinalados os resultados dos primeiros 50 tiquetaques de relógio.

Simulação	Agente Estado	Rede Marcação		Objecto Estado	Rede Marcação	Agente Estado	Rede Marcação		Objecto Estado	Rede Marcação
tiquetaque relógio	philosopher 0	philosopher 0 EATING	philosopher 0 THINKING	fork 0	fork 0 FORK	philosopher 1	philosopher 1 EATING	philosopher 1 THINKING	Fork 1	fork 1 FORK
1	a pensar	0	1	usado	0	a comer	1	0	usado	0
2	com fome	0	1	livre	1	a pensar	0	1	livre	1
3	a comer	1	0	usado	0	a pensar	0	1	usado	0
4	a pensar	0	1	livre	1	a pensar	0	1	livre	1
5	a comer	1	0	usado	0	a pensar	0	1	usado	0
6	a pensar	0	1	livre	1	a pensar	0	1	livre	1
7	a pensar	0	1	livre	1	a pensar	0	1	livre	1
8	a comer	1	0	usado	0	a pensar	0	1	usado	0
9	a comer	1	0	usado	0	com fome	0	1	usado	0
10	a comer	1	0	usado	0	com fome	0	1	usado	0
11	a pensar	0	1	usado	0	a comer	1	0	usado	0
12	com fome	0	1	livre	1	a pensar	0	1	livre	1
13	a comer	1	0	usado	0	com fome	0	1	usado	0
14	a pensar	0	1	usado	0	a comer	1	0	usado	0
15	com fome	0	1	livre	1	a pensar	0	1	livre	1
16	a comer	1	0	usado	0	a pensar	0	1	usado	0
17	a comer	1	0	usado	0	com fome	0	1	usado	0
18	a pensar	0	1	usado	0	a comer	1	0	usado	0
19	com fome	0	1	usado	0	a comer	1	0	usado	0
20	com fome	0	1	usado	0	a comer	1	0	usado	0
21	com fome	0	1	livre	1	a pensar	0	1	livre	1

22	a comer	1	0	usado	0	a pensar	0	1	usado	0
23	a comer	1	0	usado	0	a pensar	0	1	usado	0
24	a comer	1	0	usado	0	a pensar	0	1	usado	0
25	a pensar	0	1	usado	0	a comer	1	0	usado	0
26	com fome	0	1	livre	1	a pensar	0	1	livre	1
27	a comer	1	0	usado	0	a pensar	0	1	usado	0
28	a pensar	0	1	livre	1	a pensar	0	1	livre	1
29	a pensar	0	1	livre	1	a pensar	0	1	livre	1
30	a comer	1	0	usado	0	com fome	0	1	usado	0
31	a comer	1	0	usado	0	com fome	0	1	usado	0
32	a pensar	0	1	usado	0	a comer	1	0	usado	0
33	a pensar	0	1	livre	1	a pensar	0	1	livre	1
34	a pensar	0	1	usado	0	a comer	1	0	usado	0
35	com fome	0	1	usado	0	a comer	1	0	usado	0
36	com fome	0	1	usado	0	a comer	1	0	usado	0
37	com fome	0	1	usado	0	a comer	1	0	usado	0
38	com fome	0	1	livre	1	a pensar	0	1	livre	1
39	a comer	1	0	usado	0	com fome	0	1	usado	0
40	a pensar	0	1	usado	0	a comer	1	0	usado	0
41	a pensar	0	1	livre	1	a pensar	0	1	livre	1
42	a pensar	0	1	livre	1	a pensar	0	1	livre	1
43	a comer	1	0	usado	0	com fome	0	1	usado	0
44	a comer	1	0	usado	0	com fome	0	1	usado	0
45	a pensar	0	1	usado	0	a comer	1	0	usado	0
46	a pensar	0	1	usado	0	a comer	1	0	usado	0
47	a pensar	0	1	livre	1	a pensar	0	1	livre	1
48	a comer	1	0	usado	0	a pensar	0	1	usado	0
49	a pensar	0	1	livre	1	a pensar	0	1	livre	1
50	a pensar	0	1	livre	1	a pensar	0	1	livre	1

Tabela C.1. Estados de entidades e marcação de Rede de Petri em tempo real (2 filósofos)

tick	p0	p0E	p0T	f0	f0F	p1	p1E	p1T	f1	f1F	p2	p2E	p2T	f2	f2F	p3	p3E	p3T	F3	f3F	p4	P4E	p4T	f4	f4F
1	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	L	1	cf	0	1	u	0
2	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	U	0	ac	1	0	u	0
3	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0

4	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0
5	cf	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0
6	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1
7	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1
8	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
9	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0
10	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0
11	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1
12	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0
13	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
14	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
15	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1
16	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0
17	ac	1	0	u	0	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
18	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
19	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
20	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
21	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
22	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
23	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1
24	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
25	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
26	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1
27	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1
28	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1
29	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0
30	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1
31	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
32	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
33	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1
34	ac	1	0	u	0	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	u	0
35	ap	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1
36	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	ap	0	1	u	0
37	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0
38	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	ap	0	1	l	1
39	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	ap	0	1	u	0

40	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1	ap	0	1	u	0
41	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
42	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
43	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
44	cf	0	1	L	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
45	ac	1	0	U	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
46	ap	0	1	U	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
47	ap	0	1	U	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
48	cf	0	1	U	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
49	cf	0	1	L	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0
50	cf	0	1	L	1	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1

Legenda:
p[0-4] – Estado do agente philosopher [0-4]
p[0-4]E – Marcação do lugar da rede philosopher [0-4] EATING
p[0-4]T – Marcação do lugar da rede philosopher [0-4] THINKING
f[0-4] – Estado do objecto fork [0-4]
f[0-4]F – Marcação do lugar da rede fork [0-4] FORK
ap – a pensar; ac – a comer; cf – com fome
u - usado; l – livre
tick – tiquetaque de relógio

Tabela C.2. Estados de entidades e marcação de Rede de Petri em tempo real (5 filósofos)

tick	p0	p0E	p0T	f0	f0F	p1	p1E	p1T	f1	f1F	p2	p2E	p2T	f2	f2F	p3	p3E	p3T	f3	f3F	p4	P4E	p4T	f4	f4F
1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1
2	ap	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1
3	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0
4	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0
5	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0
6	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0
7	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1
8	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1
9	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	u	0
10	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0
11	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
12	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1
13	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0

14	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0
15	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	u	0
16	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1
17	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
18	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
19	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0
20	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0
21	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0
22	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0
23	cf	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0
24	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	u	0
25	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
26	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
27	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0
28	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0
29	ac	1	0	u	0	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
30	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1
31	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
32	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
33	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
34	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
35	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
36	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0
37	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	cf	0	1	u	0
38	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1
39	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
40	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
41	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
42	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
43	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
44	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
45	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1
46	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
47	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1
48	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
49	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0

50	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0

Legenda:
p[0-4] – Estado do agente philosopher [0-4]
p[0-4]E – Marcação do lugar da rede philosopher [0-4] EATING
p[0-4]T – Marcação do lugar da rede philosopher [0-4] THINKING
f[0-4] – Estado do objecto fork [0-4]
f[0-4]F – Marcação do lugar da rede fork [0-4] FORK
ap – a pensar; ac – a comer; cf – com fome
u - usado; l – livre
tick – tiquetaque de relógio

Tabela C.3. Estados de entidades e marcação de Rede de Petri em tempo real (10 filósofos – Do filósofo 0 ao filósofo 4)

tick	p5	p5E	p5T	f5	f5F	p6	p6E	p6T	f6	f6F	p7	p7E	p7T	f7	f7F	p8	p8E	p8T	f8	f8F	p9	p9E	p9T	f9	f9F
1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
2	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0
3	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0
4	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1
5	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	u	0
6	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1
7	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1
8	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1
9	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1
10	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1	ap	0	1	u	0
11	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0
12	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1
13	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1
14	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	u	0
15	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
16	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
17	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
18	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
19	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
20	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
21	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
22	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0
23	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0

24	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1
25	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0
26	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0
27	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1
28	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1
29	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
30	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
31	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	ap	0	1	l	1	ap	0	1	l	1
32	cf	0	1	l	1	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1
33	ac	1	0	u	0	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
34	ac	1	0	u	0	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
35	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
36	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
37	ac	1	0	u	0	cf	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
38	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0
39	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
40	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
41	ap	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0
42	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	u	0	ac	1	0	u	0
43	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	l	1
44	ap	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1
45	cf	0	1	l	1	ap	0	1	l	1	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
46	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0
47	cf	0	1	l	1	ap	0	1	u	0	ac	1	0	u	0	ap	0	1	u	0	ac	1	0	u	0
48	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	ap	0	1	l	1	ap	0	1	l	1
49	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0
50	ac	1	0	u	0	cf	0	1	u	0	ac	1	0	u	0	cf	0	1	l	1	ap	0	1	l	1

Legenda:
p[5-9] – Estado do agente philosopher [5-9]
p[5-9]E – Marcação do lugar da rede philosopher [5-9] EATING
p[5-9]T – Marcação do lugar da rede philosopher [5-9] THINKING
f[5-9] – Estado do objecto fork [5-9]
f[5-9]F – Marcação do lugar da rede fork [5-9] FORK
ap – a pensar; ac – a comer; cf – com fome
u - usado; l – livre
tick – tiquetaque de relógio

Tabela C.4. Estados de entidades e marcação de Rede de Petri em tempo real (10 filósofos – Do filósofo 5 ao filósofo 9)

As tabelas C.5, C.6 e C.7 exibem resultados da marcação da Rede de frequência de lugares marcados. Foram executadas 50 simulações do Jantar com 3 filósofos (Tabela C.5), 4 filósofos (Tabela C.6) e 5 filósofos (Tabela C.7). Para cada simulação, foram assinalados os resultados ao fim de 250 tiquetaques de relógio.

Simulação	philosopher EATING			philosopher THINKING			fork FORK		
Número	0	1	2	0	1	2	0	1	2
1	82	60	66	168	190	184	108	124	102
2	64	71	79	186	179	171	115	100	107
3	54	82	73	196	168	177	114	95	123
4	60	84	66	190	166	184	106	100	124
5	55	81	70	195	169	180	114	99	125
6	83	78	55	167	172	195	89	117	112
7	68	64	82	182	186	168	118	104	100
8	81	65	65	169	185	185	104	120	104
9	68	66	72	182	184	178	116	112	110
10	65	71	79	185	179	171	114	100	106
11	86	68	61	164	182	189	96	121	103
12	70	76	65	180	174	185	104	109	115
13	68	86	60	182	164	190	96	104	122
14	80	63	68	170	187	182	107	119	102
15	62	84	61	188	166	189	104	105	127
16	54	77	85	196	173	165	119	88	111
17	56	89	64	194	161	186	105	97	130
18	75	74	65	175	176	185	101	111	110
19	65	65	78	185	185	172	120	107	107
20	69	70	70	181	180	180	111	110	111
21	68	72	70	182	178	180	110	108	112
22	63	80	73	187	170	177	107	97	114
23	74	75	60	176	175	190	101	115	116
24	65	83	68	185	167	182	102	99	117
25	75	65	74	175	185	176	110	111	101
26	71	73	68	179	177	182	106	109	111
27	75	67	69	175	183	181	108	114	106
28	71	59	84	179	191	166	120	107	95
29	75	70	68	175	180	182	105	112	107
30	78	65	70	172	185	180	107	115	102
31	58	74	77	192	176	173	118	99	115
32	68	72	67	182	178	183	110	111	115
33	64	76	71	186	174	179	110	103	115
34	69	68	75	181	182	175	113	107	106
35	67	66	79	183	184	171	117	105	104
36	62	77	72	188	173	178	111	101	116
37	72	63	72	178	187	178	115	115	106
38	77	65	73	173	185	177	108	112	100
39	69	69	70	181	181	180	112	111	111
40	79	65	66	171	185	184	106	119	105
41	73	63	77	177	187	173	114	110	100

42	77	78	56	173	172	194	95	116	117
43	66	75	71	184	175	179	109	104	113
44	72	62	78	178	188	172	116	110	100
45	74	70	66	176	180	184	106	114	110
46	69	72	72	181	178	178	109	106	109
47	70	66	70	180	184	180	114	114	110
48	76	65	68	174	185	182	109	117	106
49	61	68	70	189	182	180	121	112	119
50	73	77	60	177	173	190	100	113	117

Tabela C.5. Resultados de Rede de frequência de lugares marcados (3 filósofos)

Simulação	philosopher EATING				philosopher THINKING				fork FORK			
Número	0	1	2	3	0	1	2	3	0	1	2	3
1	65	105	67	81	185	145	183	169	80	78	102	104
2	92	68	90	78	158	182	160	172	90	92	82	80
3	76	84	81	86	174	166	169	164	90	85	83	88
4	78	91	75	81	172	159	175	169	81	84	94	91
5	91	77	85	73	159	173	165	177	82	88	92	86
6	80	74	93	68	170	176	157	182	96	83	89	102
7	80	73	97	80	170	177	153	170	97	80	73	90
8	76	84	72	89	174	166	178	161	90	94	89	85
9	69	97	72	91	181	153	178	159	84	81	87	90
10	85	82	83	73	165	168	167	177	83	85	94	92
11	83	78	99	76	167	172	151	174	89	73	75	91
12	99	86	73	78	151	164	177	172	65	91	99	73
13	86	80	82	88	164	170	168	162	84	88	80	76
14	94	71	94	72	156	179	156	178	85	85	84	84
15	82	77	79	75	168	173	171	175	91	94	96	93
16	73	94	73	97	177	156	177	153	83	83	80	80
17	72	81	77	89	178	169	173	161	97	92	84	89
18	80	85	79	86	170	165	171	164	85	86	85	84
19	74	90	76	93	176	160	174	157	86	84	81	83
20	90	71	82	81	160	179	168	169	89	97	87	79
21	85	83	81	75	165	167	169	175	82	86	94	90
22	70	82	78	84	180	168	172	166	98	90	88	96
23	82	81	79	81	168	169	171	169	87	90	90	87
24	94	71	90	70	156	179	160	180	85	89	90	86
25	81	78	85	79	169	172	165	171	91	87	86	90
26	93	68	91	74	157	182	159	176	89	91	85	83
27	87	75	88	73	163	175	162	177	88	87	89	90
28	84	77	82	79	166	173	168	171	89	91	89	87
29	68	82	79	86	182	168	171	164	100	89	85	96
30	67	102	67	93	183	148	183	157	81	81	90	90
31	80	91	64	79	170	159	186	171	79	95	107	91
32	79	82	78	86	171	168	172	164	89	90	86	85
33	79	69	89	87	171	181	161	163	102	92	74	84
34	70	87	80	83	180	163	170	167	93	83	87	97
35	69	83	77	80	181	167	173	170	98	90	93	101
36	79	84	74	92	171	166	176	158	87	92	84	79

	37	81	76	85	74	169	174	165	176	93	89	91	95
	38	74	89	76	91	176	161	174	159	87	85	83	85
	39	82	72	79	85	168	178	171	165	96	99	86	83
	40	68	94	72	93	182	156	178	157	88	84	85	89
	41	93	71	82	70	157	179	168	180	86	97	98	87
	42	91	71	101	64	159	179	149	186	88	78	85	95
	43	67	92	73	87	183	158	177	163	91	85	90	96
	44	78	83	81	84	172	167	169	166	89	86	85	88
	45	90	74	89	59	160	176	161	191	86	87	102	101
	46	76	81	77	85	174	169	173	165	93	92	88	89
	47	71	79	84	84	179	171	166	166	100	87	82	95
	48	81	81	89	81	169	169	161	169	88	80	80	88
	49	82	78	92	73	168	172	158	177	90	80	85	95
	50	73	97	80	82	177	153	170	168	80	73	88	95

Tabela C.6. Resultados de Rede de frequência de lugares marcados (4 filósofos)

Simulação	philosopher EATING					philosopher THINKING					fork FORK				
Número	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
1	89	70	76	87	64	161	180	174	163	186	91	104	87	99	97
2	81	87	56	96	72	169	163	194	154	178	82	107	98	82	97
3	86	79	70	78	73	164	171	180	172	177	85	101	102	99	91
4	86	73	75	78	79	164	177	175	172	171	91	102	97	93	85
5	63	102	61	93	67	187	148	189	157	183	85	87	96	90	120
6	85	73	74	89	66	165	177	176	161	184	92	103	87	95	99
7	75	89	69	72	79	175	161	181	178	171	86	92	109	99	96
8	92	66	92	76	73	158	184	158	174	177	92	92	82	101	85
9	81	75	77	81	72	169	175	173	169	178	94	98	92	97	97
10	80	72	73	73	80	170	178	177	177	170	98	105	104	97	90
11	86	74	80	89	65	164	176	170	161	185	90	96	81	96	99
12	66	88	72	83	81	184	162	178	167	169	96	90	95	86	103
13	62	75	88	69	102	188	175	162	181	148	113	87	93	79	86
14	82	71	80	75	78	168	179	170	175	172	97	99	95	97	90
15	82	80	77	85	71	168	170	173	165	179	88	93	88	94	97
16	72	75	87	70	91	178	175	163	180	159	103	88	93	89	87
17	77	72	88	69	82	173	178	162	181	168	101	90	93	99	91
18	71	83	77	75	79	179	167	173	175	171	96	90	98	96	100
19	90	59	99	59	82	160	191	151	191	168	101	92	92	109	78
20	73	85	76	79	79	177	165	174	171	171	92	89	95	92	98
21	83	77	71	87	69	167	173	179	163	181	90	102	92	94	98
22	99	56	93	64	69	151	194	157	186	181	95	101	93	117	82
23	70	74	79	77	81	180	176	171	173	169	106	97	94	92	99
24	77	73	81	78	77	173	177	169	172	173	100	96	91	95	96
25	97	74	78	80	65	153	176	172	170	185	79	98	92	105	88
26	73	68	75	90	68	177	182	175	160	182	109	107	85	92	109
27	75	86	72	73	76	175	164	178	177	174	89	92	105	101	99
28	81	81	66	87	70	169	169	184	163	180	88	103	97	93	99
29	91	70	94	64	72	159	180	156	186	178	89	86	92	114	87
30	77	70	83	78	77	173	180	167	172	173	103	97	89	95	96
31	75	78	96	70	78	175	172	154	180	172	97	76	84	102	97

32	80	81	78	63	86	170	169	172	187	164	89	91	109	101	84
33	82	68	88	68	80	168	182	162	182	170	100	94	94	102	88
34	78	80	79	77	74	172	170	171	173	176	92	91	94	99	98
35	76	76	77	65	97	174	174	173	185	153	98	97	108	88	77
36	85	63	82	79	73	165	187	168	171	177	102	105	89	98	92
37	79	76	70	82	81	171	174	180	168	169	95	104	98	87	90
38	88	73	70	97	66	162	177	180	153	184	89	107	83	87	96
39	78	66	83	77	79	172	184	167	173	171	106	101	90	94	93
40	79	84	69	95	64	171	166	181	155	186	87	97	86	91	107
41	79	82	75	86	69	171	168	175	164	181	89	93	89	95	102
42	76	69	98	63	89	174	181	152	187	161	105	83	89	98	85
43	80	58	103	67	81	170	192	147	183	169	112	89	80	102	89
44	91	67	91	75	75	159	183	159	175	175	92	92	84	100	84
45	85	75	79	75	74	165	175	171	175	176	90	96	96	101	91
46	61	93	76	67	86	189	157	174	183	164	96	81	107	97	103
47	88	70	86	81	67	162	180	164	169	183	92	94	83	102	95
48	79	70	96	66	78	171	180	154	184	172	101	84	88	106	93
49	62	87	73	73	96	188	163	177	177	154	101	90	104	81	92
50	75	84	67	97	62	175	166	183	153	188	91	99	86	91	113

Tabela C.7. Resultados de Rede de frequência de lugares marcados (5 filósofos)

As tabelas C.8, C.9 e C.10 exibem resultados da marcação da Rede de frequência de disparo de transições. Foram executadas 50 simulações do Jantar com 3 filósofos (Tabela C.8), 4 filósofos (Tabela C.9) e 5 filósofos (Tabela C.10). Para cada simulação, foram assinalados os resultados ao fim de 250 tiquetaques de relógio.

Simulação	philosopher ToEAT			philosopher ToTHINK		
Número	0	1	2	0	1	2
1	38	34	38	38	33	38
2	39	37	37	39	37	36
3	36	34	31	36	34	31
4	34	34	35	34	33	35
5	35	34	35	35	34	35
6	34	38	37	34	37	37
7	32	31	30	32	30	30
8	38	40	35	38	40	34
9	36	35	34	35	35	34
10	34	34	37	33	34	37
11	36	36	36	36	36	35
12	36	35	36	35	35	36
13	35	35	37	34	35	37
14	35	36	33	35	36	32
15	32	33	35	32	33	34
16	33	34	34	33	34	34
17	35	36	36	34	36	36
18	33	35	34	33	34	34
19	34	33	37	34	33	36

20	37	39	38	37	39	37
21	37	36	34	37	35	34
22	39	37	38	39	36	38
23	36	34	37	36	34	36
24	36	36	35	35	36	35
25	33	34	35	33	34	34
26	36	37	35	35	37	35
27	35	36	36	35	36	35
28	31	34	33	31	34	32
29	34	33	33	34	32	33
30	39	37	40	39	37	39
31	35	35	35	35	34	35
32	32	36	35	32	36	35
33	36	35	36	36	35	35
34	38	36	35	37	36	35
35	39	36	36	39	35	36
36	36	31	32	35	31	32
37	39	42	41	39	42	40
38	38	34	32	37	34	32
39	35	35	35	35	35	34
40	36	39	38	36	39	38
41	39	34	36	39	33	36
42	37	35	35	37	34	35
43	36	37	36	36	36	36
44	34	35	33	34	35	32
45	39	36	36	39	35	36
46	35	33	34	35	32	34
47	38	35	37	38	35	37
48	32	37	35	32	36	35
49	42	40	36	42	40	35
50	36	33	35	35	33	35

Tabela C.8. Resultados de Rede de frequência de disparo de transições (3 filósofos)

Simulação	philosopher ToEAT				philosopher ToTHINK			
Número	0	1	2	3	0	1	2	3
1	34	34	37	40	34	33	37	39
2	38	37	43	44	37	37	42	44
3	41	45	45	43	41	45	45	43
4	42	37	45	35	41	37	44	35
5	32	42	36	43	32	41	36	43
6	42	45	45	40	42	44	45	39
7	40	43	36	41	39	43	35	41
8	42	40	41	41	42	40	40	41
9	40	36	44	33	40	36	44	32
10	38	37	38	39	37	37	38	39
11	40	39	43	39	40	38	43	39
12	40	40	42	42	40	39	42	41
13	40	42	41	42	40	42	41	41

14	40	41	38	46	40	40	38	45
15	43	40	40	37	43	39	40	36
16	44	45	38	42	43	45	37	42
17	38	38	37	38	38	38	36	38
18	40	40	40	42	40	40	39	42
19	44	35	44	35	44	35	43	35
20	43	43	44	38	42	43	43	38
21	43	39	39	39	42	39	38	39
22	37	42	41	41	37	41	41	40
23	36	38	34	43	35	38	34	43
24	38	49	39	43	37	49	38	43
25	43	37	40	37	43	36	40	36
26	34	35	41	45	34	35	41	45
27	37	34	40	37	37	34	39	37
28	38	42	42	37	38	42	42	37
29	44	41	35	45	44	41	35	44
30	42	41	42	39	42	41	42	38
31	34	50	34	47	34	49	34	46
32	43	44	45	37	42	44	45	37
33	42	39	43	44	42	38	43	43
34	47	33	40	32	46	33	39	32
35	38	40	40	40	38	39	40	39
36	34	37	35	38	34	36	35	37
37	40	41	40	42	39	41	39	42
38	40	34	39	43	40	33	39	43
39	41	37	34	41	41	36	34	40
40	43	30	38	36	42	30	38	36
41	38	40	36	41	38	40	36	41
42	39	41	45	43	39	40	45	42
43	46	38	41	40	45	38	40	40
44	38	40	40	33	37	40	39	33
45	38	42	41	34	38	41	41	33
46	38	40	42	44	37	40	42	44
47	42	42	39	41	42	42	38	41
48	45	41	34	38	45	41	34	38
49	34	43	35	40	34	42	35	39
50	39	39	38	45	39	38	38	44

Tabela C.9. Resultados de Rede de frequência de disparo de transições (4 filósofos)

Simulação	philosopher ToEAT					philosopher ToTHINK				
Número	0	1	2	3	4	0	1	2	3	4
1	34	36	45	44	40	33	36	44	44	40
2	38	40	35	36	39	38	40	34	36	38
3	39	40	38	38	39	39	40	37	38	38
4	31	42	39	36	43	31	42	39	36	42
5	40	41	39	36	40	39	41	39	36	40
6	37	46	31	45	42	37	46	30	45	42
7	39	47	39	38	37	39	47	38	38	36
8	41	43	39	40	37	40	43	39	39	37

9	40	36	37	38	42	40	36	37	38	42
10	38	37	41	41	38	38	36	41	41	38
11	39	38	37	46	40	39	37	37	46	39
12	38	38	35	38	34	37	38	34	38	34
13	45	38	42	42	45	44	38	41	42	45
14	41	38	42	39	42	41	38	42	39	42
15	41	40	38	41	44	41	39	38	41	44
16	36	35	36	36	39	36	35	36	35	39
17	34	36	37	38	33	34	36	36	38	33
18	39	43	38	38	36	39	43	38	37	36
19	38	38	39	38	40	38	37	39	38	40
20	38	35	37	38	37	37	35	37	38	37
21	39	40	46	42	39	39	40	45	42	38
22	40	40	33	40	44	40	40	33	40	43
23	41	41	40	42	40	41	40	40	42	39
24	39	41	39	44	36	39	40	39	43	36
25	42	34	36	38	38	42	34	35	38	38
26	42	42	37	39	41	42	42	36	39	41
27	34	40	33	36	38	34	40	32	36	37
28	35	35	37	35	45	34	35	37	34	45
29	34	38	32	36	39	34	37	32	35	39
30	40	38	40	40	40	40	38	39	40	39
31	42	42	39	38	43	42	41	39	38	43
32	43	34	33	37	40	43	33	33	37	39
33	43	42	42	42	40	43	41	42	42	39
34	38	45	37	37	37	38	44	37	37	37
35	36	40	35	38	35	36	39	35	38	34
36	42	33	39	37	38	42	32	39	37	38
37	42	37	43	44	36	41	37	43	44	36
38	36	37	37	39	36	36	36	37	39	36
39	41	37	39	34	33	41	37	38	34	32
40	38	40	38	42	34	38	40	38	42	34
41	37	40	38	37	38	36	40	37	37	38
42	37	39	39	44	33	36	39	39	43	33
43	39	42	42	43	41	38	42	42	42	41
44	42	37	38	37	37	42	37	38	37	36
45	39	43	40	41	39	38	43	40	41	39
46	37	37	37	45	37	37	37	37	44	37
47	40	37	36	45	40	40	37	35	45	40
48	35	39	39	39	44	35	39	39	39	43
49	34	36	34	36	39	34	35	34	35	39
50	36	40	42	38	40	36	39	42	38	40

Tabela C.10. Resultados de Rede de frequência de disparo de transições (5 filósofos)

As tabelas C.11 a C.15 apresentam resultados de simulações efectuadas com o modelo de Jantar com o filosofo glutão.

A tabela C.11. regista o tiquetaque de relógio no qual se verificou o impasse nas 50 simulações efectuadas para modelos com 2, 5 e 8 filósofos.

A tabela C.12 exibe resultados da Rede de frequência de lugares marcados de 50 simulações, aquando da ocorrência do impasse. A tabela C.13 exibe resultados da Rede de frequência de lugares marcados das mesmas 50 simulações, ao fim de 300 tiquetaques de relógio.

A tabela C.14 exibe resultados da Rede de frequência de disparo de transições de 50 simulações, aquando da ocorrência do impasse. A tabela C.15 exibe resultados da Rede de frequência de disparo de transições das mesmas 50 simulações, ao fim de 300 tiquetaques de relógio.

Simulação	Impasse - Número filósofos		
Número	2	5	8
1	4	18	184
2	7	14	109
3	11	21	255
4	4	8	33
5	5	85	101
6	7	12	25
7	17	3	100
8	11	36	278
9	9	20	110
10	4	24	242
11	1	18	162
12	3	33	15
13	4	16	362
14	9	49	1122
15	4	14	14
16	5	25	609
17	3	11	21
18	3	35	48
19	3	27	103
20	9	41	90
21	5	95	79
22	4	13	414
23	5	11	428
24	3	27	625
25	6	19	186
26	3	16	5
27	3	32	540
28	1	39	389
29	6	41	53
30	1	8	414
31	10	23	39
32	7	30	399
33	4	9	87
34	4	51	169

35	5	6	473
36	3	21	160
37	4	25	17
38	6	7	211
39	9	25	77
40	8	38	46
41	3	45	212
42	3	6	68
43	1	35	739
44	3	28	126
45	5	19	624
46	6	7	33
47	11	27	535
48	8	11	377
49	6	65	98
50	3	15	21

Tabela C.11. Ocorrência de impasses no modelo com filósofo glutão

Simulação	Impasse	philosopher EATING						philosopher THINKING						fork FORK				
Número	tiquetaque	0	1	2	3	4	G	0	1	2	3	4	G	0	1	2	3	4
1	18	5	5	4	5	3	1	13	13	14	13	15	17	7	8	8	9	9
2	14	2	7	3	1	6	1	12	7	11	13	8	13	4	3	9	6	5
3	21	3	10	4	4	10	1	18	11	17	17	11	20	7	6	12	6	7
4	8	1	1	3	1	3	1	7	7	5	7	5	7	5	3	3	3	3
5	85	26	28	26	27	29	1	59	57	59	58	56	84	30	30	31	28	29
6	12	3	4	4	1	4	1	9	8	8	11	8	11	4	3	6	6	4
7	3	0	0	0	1	0	1	3	3	3	2	3	2	2	2	1	1	2
8	36	9	9	11	11	11	1	27	27	25	25	25	35	17	15	13	13	15
9	20	2	9	4	8	4	1	18	11	16	12	16	19	8	6	7	7	13
10	24	3	9	6	6	10	1	21	15	18	18	14	23	11	8	11	7	10
11	18	3	5	6	4	8	1	15	13	12	14	10	17	9	6	7	5	6
12	33	6	14	7	12	14	1	27	19	26	21	19	32	12	11	13	6	12
13	16	3	8	3	4	6	1	13	8	13	12	10	15	4	4	8	5	6
14	49	11	17	15	14	19	1	38	32	34	35	30	48	20	16	19	15	18
15	14	6	2	6	2	5	1	8	12	8	12	9	13	5	5	5	6	2
16	25	5	6	7	7	10	1	20	19	18	18	15	24	13	11	10	7	9
17	11	3	2	4	1	4	1	8	9	7	10	7	10	5	4	5	5	3
18	35	6	11	10	10	16	1	29	24	25	25	19	34	17	13	14	8	12
19	27	7	10	9	12	5	1	20	17	18	15	22	26	9	7	5	9	14
20	41	15	10	11	13	12	1	26	31	30	28	29	40	15	19	16	15	13
21	95	32	29	30	31	29	1	63	66	65	64	66	94	33	35	33	34	33
22	13	3	2	7	1	5	1	10	11	6	12	8	12	7	3	4	6	4
23	11	2	3	3	2	5	1	9	8	8	9	6	10	5	4	5	3	3
24	27	6	7	7	6	11	1	21	20	20	21	16	26	13	12	13	9	9
25	19	3	7	5	4	10	1	16	12	14	15	9	18	8	6	9	4	5
26	16	8	3	3	5	2	1	8	13	13	11	14	15	4	9	7	8	5
27	32	7	11	7	13	10	1	25	21	25	19	22	31	13	13	11	8	14
28	39	18	8	15	14	6	1	21	31	24	25	33	38	12	15	9	18	14
29	41	12	9	19	13	12	1	29	32	22	28	29	40	19	12	8	15	16

30	8	1	2	3	0	4	1	7	6	5	8	4	7	4	2	4	3	2
31	23	8	8	6	5	5	1	15	15	17	18	18	22	6	8	11	12	9
32	30	10	6	9	11	8	1	20	24	21	19	22	29	13	14	9	10	11
33	9	2	3	2	2	2	1	7	6	7	7	7	8	3	3	4	4	4
34	51	14	21	9	19	19	1	37	30	42	32	32	50	15	20	22	12	17
35	6	0	1	3	0	4	1	6	5	3	6	2	5	4	1	2	1	1
36	21	5	10	2	9	7	1	16	11	19	12	14	20	5	8	9	4	8
37	25	4	11	6	8	9	1	21	14	19	17	16	24	9	7	10	7	11
38	7	1	1	3	1	4	1	6	6	4	6	3	6	4	2	2	1	1
39	25	5	7	7	8	7	1	20	18	18	17	18	24	12	10	9	9	12
40	38	10	14	9	8	14	1	28	24	29	30	24	37	13	14	20	15	13
41	45	12	14	12	14	17	1	33	31	33	31	28	44	18	18	18	13	15
42	6	0	1	1	0	3	1	6	5	5	6	3	5	4	3	4	2	2
43	35	15	11	7	15	9	1	20	24	28	20	26	34	8	16	12	10	10
44	28	8	10	3	13	6	1	20	18	25	15	22	27	9	14	11	8	13
45	19	4	3	8	2	9	1	15	16	11	17	10	18	11	7	8	7	5
46	7	0	2	2	1	4	1	7	5	5	6	3	6	4	2	3	1	2
47	27	7	6	11	10	5	1	20	21	16	17	22	26	13	9	5	11	14
48	11	5	1	5	2	2	1	6	10	6	9	9	10	4	4	3	6	3
49	65	14	27	21	15	23	1	51	38	44	50	42	64	23	16	28	26	27
50	15	5	2	5	5	3	1	10	13	10	10	12	14	7	7	4	6	6

Tabela C.12. Resultados de Rede de frequência de lugares marcados até ao impasse (5 filósofos com glutão)

Simulação		philosopher EATING						philosopher THINKING						fork FORK				
Número	tique	0	1	2	3	4	G	0	1	2	3	4	G	0	1	2	3	4
1	300	5	5	4	5	3	283	295	295	296	295	297	17	7	8	8	9	9
2	300	2	7	3	1	6	287	298	293	297	299	294	13	4	3	9	6	5
3	300	3	10	4	4	10	280	297	290	296	296	290	20	7	6	12	6	7
4	300	1	1	3	1	3	293	299	299	297	299	297	7	5	3	3	3	3
5	300	26	28	26	27	29	216	274	272	274	273	271	84	30	30	31	28	29
6	300	3	4	4	1	4	289	297	296	296	299	296	11	4	3	6	6	4
7	300	0	0	0	1	0	298	300	300	300	299	300	2	2	2	1	1	2
8	300	9	9	11	11	11	265	291	291	289	289	289	35	17	15	13	13	15
9	300	2	9	4	8	4	281	298	291	296	292	296	19	8	6	7	7	13
10	300	3	9	6	6	10	277	297	291	294	294	290	23	11	8	11	7	10
11	300	3	5	6	4	8	283	297	295	294	296	292	17	9	6	7	5	6
12	300	6	14	7	12	14	268	294	286	293	288	286	32	12	11	13	6	12
13	300	3	8	3	4	6	285	297	292	297	296	294	15	4	4	8	5	6
14	300	11	17	15	14	19	252	289	283	285	286	281	48	20	16	19	15	18
15	300	6	2	6	2	5	287	294	298	294	298	295	13	5	5	5	6	2
16	300	5	6	7	7	10	276	295	294	293	293	290	24	13	11	10	7	9
17	300	3	2	4	1	4	290	297	298	296	299	296	10	5	4	5	5	3
18	300	6	11	10	10	16	266	294	289	290	290	284	34	17	13	14	8	12
19	300	7	10	9	12	5	274	293	290	291	288	295	26	9	7	5	9	14
20	300	15	10	11	13	12	260	285	290	289	287	288	40	15	19	16	15	13
21	300	32	29	30	31	29	206	268	271	270	269	271	94	33	35	33	34	33
22	300	3	2	7	1	5	288	297	298	293	299	295	12	7	3	4	6	4
23	300	2	3	3	2	5	290	298	297	297	298	295	10	5	4	5	3	3

24	300	6	7	7	6	11	274	294	293	293	294	289	26	13	12	13	9	9
25	300	3	7	5	4	10	282	297	293	295	296	290	18	8	6	9	4	5
26	300	8	3	3	5	2	285	292	297	297	295	298	15	4	9	7	8	5
27	300	7	11	7	13	10	269	293	289	293	287	290	31	13	13	11	8	14
28	300	18	8	15	14	6	262	282	292	285	286	294	38	12	15	9	18	14
29	300	12	9	19	13	12	260	288	291	281	287	288	40	19	12	8	15	16
30	300	1	2	3	0	4	293	299	298	297	300	296	7	4	2	4	3	2
31	300	8	8	6	5	5	278	292	292	294	295	295	22	6	8	11	12	9
32	300	10	6	9	11	8	271	290	294	291	289	292	29	13	14	9	10	11
33	300	2	3	2	2	2	292	298	297	298	298	298	8	3	3	4	4	4
34	300	14	21	9	19	19	250	286	279	291	281	281	50	15	20	22	12	17
35	300	0	1	3	0	4	295	300	299	297	300	296	5	4	1	2	1	1
36	300	5	10	2	9	7	280	295	290	298	291	293	20	5	8	9	4	8
37	300	4	11	6	8	9	276	296	289	294	292	291	24	9	7	10	7	11
38	300	1	1	3	1	4	294	299	299	297	299	296	6	4	2	2	1	1
39	300	5	7	7	8	7	276	295	293	293	292	293	24	12	10	9	9	12
40	300	10	14	9	8	14	263	290	286	291	292	286	37	13	14	20	15	13
41	300	12	14	12	14	17	256	288	286	288	286	283	44	18	18	18	13	15
42	300	0	1	1	0	3	295	300	299	299	300	297	5	4	3	4	2	2
43	300	15	11	7	15	9	266	285	289	293	285	291	34	8	16	12	10	10
44	300	8	10	3	13	6	273	292	290	297	287	294	27	9	14	11	8	13
45	300	4	3	8	2	9	282	296	297	292	298	291	18	11	7	8	7	5
46	300	0	2	2	1	4	294	300	298	298	299	296	6	4	2	3	1	2
47	300	7	6	11	10	5	274	293	294	289	290	295	26	13	9	5	11	14
48	300	5	1	5	2	2	290	295	299	295	298	298	10	4	4	3	6	3
49	300	14	27	21	15	23	236	286	273	279	285	277	64	23	16	28	26	27
50	300	5	2	5	5	3	286	295	298	295	295	297	14	7	7	4	6	6

Tabela C.13. Resultados de Rede de frequência de lugares marcados até 300 tiquetaques de relógio (5 filósofos com glutão)

Simulação	Impasse	philosopher ToEAT						philosopher ToTHINK				
Número	Tiquetaque	0	1	2	3	4	G	0	1	2	3	4
1	18	3	2	2	4	3	1	3	2	2	4	3
2	14	1	2	3	1	4	1	1	2	3	1	4
3	21	3	3	3	2	3	1	3	3	3	2	3
4	8	1	1	1	1	2	1	1	1	1	1	2
5	85	12	16	11	12	12	1	12	16	11	12	12
6	12	2	2	1	1	2	1	2	2	1	1	2
7	3	0	0	0	1	0	1	0	0	0	1	0
8	36	6	6	6	5	5	1	6	6	6	5	5
9	20	2	4	2	2	4	1	2	4	2	2	4
10	24	2	4	3	3	4	1	2	4	3	3	4
11	18	2	4	3	3	4	1	2	4	3	3	4
12	33	5	7	6	4	4	1	5	7	6	4	4
13	16	2	3	2	2	3	1	2	3	2	2	3
14	49	7	9	8	5	9	1	7	9	8	5	9
15	14	2	1	2	1	2	1	2	1	2	1	2
16	25	3	5	5	3	3	1	3	5	5	3	3
17	11	1	1	2	1	3	1	1	1	2	1	3

18	35	4	9	6	3	5	1	4	9	6	3	5
19	27	3	4	4	3	3	1	3	4	4	3	3
20	41	6	7	8	7	5	1	6	7	8	7	5
21	95	14	14	17	14	11	1	14	14	17	14	11
22	13	1	2	2	1	3	1	1	2	2	1	3
23	11	1	1	1	2	2	1	1	1	1	2	2
24	27	3	6	7	4	4	1	3	6	7	4	4
25	19	2	2	3	1	2	1	2	2	3	1	2
26	16	2	2	3	3	1	1	2	2	3	3	1
27	32	4	7	4	5	6	1	4	7	4	5	6
28	39	5	5	6	7	4	1	5	5	6	7	4
29	41	7	5	4	4	8	1	7	5	4	4	8
30	8	1	1	2	0	2	1	1	1	2	0	2
31	23	4	5	5	4	5	1	4	5	5	4	5
32	30	4	5	5	5	3	1	4	5	5	5	3
33	9	1	2	2	2	2	1	1	2	2	2	2
34	51	6	10	8	6	7	1	6	10	8	6	7
35	6	0	1	1	0	1	1	0	1	1	0	1
36	21	4	3	2	3	3	1	4	3	2	3	3
37	25	3	4	4	5	5	1	3	4	4	5	5
38	7	1	1	1	1	1	1	1	1	1	1	1
39	25	3	3	3	4	5	1	3	3	3	4	5
40	38	6	8	7	6	7	1	6	8	7	6	7
41	45	8	9	6	6	7	1	8	9	6	6	7
42	6	0	1	1	0	2	1	0	1	1	0	2
43	35	4	5	6	7	3	1	4	5	6	7	3
44	28	5	6	3	5	3	1	5	6	3	5	3
45	19	2	3	5	2	3	1	2	3	5	2	3
46	7	0	1	1	1	1	1	0	1	1	1	1
47	27	6	5	3	4	4	1	6	5	3	4	4
48	11	2	1	3	2	2	1	2	1	3	2	2
49	65	9	10	8	9	13	1	9	10	8	9	13
50	15	4	2	2	2	3	1	4	2	2	2	3

Tabela C.14. Resultados de Rede de frequência de disparo de transições até ao impasse (5 filósofos com glutão)

Simulação	Relógio	philosopher ToEAT						philosopher ToTHINK				
Número	Tiquetaque	0	1	2	3	4	G	0	1	2	3	4
1	300	3	2	2	4	3	1	3	2	2	4	3
2	300	1	2	3	1	4	1	1	2	3	1	4
3	300	3	3	3	2	3	1	3	3	3	2	3
4	300	1	1	1	1	2	1	1	1	1	1	2
5	300	12	16	11	12	12	1	12	16	11	12	12
6	300	2	2	1	1	2	1	2	2	1	1	2
7	300	0	0	0	1	0	1	0	0	0	1	0
8	300	6	6	6	5	5	1	6	6	6	5	5
9	300	2	4	2	2	4	1	2	4	2	2	4
10	300	2	4	3	3	4	1	2	4	3	3	4
11	300	2	4	3	3	4	1	2	4	3	3	4

12	300	5	7	6	4	4	1	5	7	6	4	4
13	300	2	3	2	2	3	1	2	3	2	2	3
14	300	7	9	8	5	9	1	7	9	8	5	9
15	300	2	1	2	1	2	1	2	1	2	1	2
16	300	3	5	5	3	3	1	3	5	5	3	3
17	300	1	1	2	1	3	1	1	1	2	1	3
18	300	4	9	6	3	5	1	4	9	6	3	5
19	300	3	4	4	3	3	1	3	4	4	3	3
20	300	6	7	8	7	5	1	6	7	8	7	5
21	300	14	14	17	14	11	1	14	14	17	14	11
22	300	1	2	2	1	3	1	1	2	2	1	3
23	300	1	1	1	2	2	1	1	1	1	2	2
24	300	3	6	7	4	4	1	3	6	7	4	4
25	300	2	2	3	1	2	1	2	2	3	1	2
26	300	2	2	3	3	1	1	2	2	3	3	1
27	300	4	7	4	5	6	1	4	7	4	5	6
28	300	5	5	6	7	4	1	5	5	6	7	4
29	300	7	5	4	4	8	1	7	5	4	4	8
30	300	1	1	2	0	2	1	1	1	2	0	2
31	300	4	5	5	4	5	1	4	5	5	4	5
32	300	4	5	5	5	3	1	4	5	5	5	3
33	300	1	2	2	2	2	1	1	2	2	2	2
34	300	6	10	8	6	7	1	6	10	8	6	7
35	300	0	1	1	0	1	1	0	1	1	0	1
36	300	4	3	2	3	3	1	4	3	2	3	3
37	300	3	4	4	5	5	1	3	4	4	5	5
38	300	1	1	1	1	1	1	1	1	1	1	1
39	300	3	3	3	4	5	1	3	3	3	4	5
40	300	6	8	7	6	7	1	6	8	7	6	7
41	300	8	9	6	6	7	1	8	9	6	6	7
42	300	0	1	1	0	2	1	0	1	1	0	2
43	300	4	5	6	7	3	1	4	5	6	7	3
44	300	5	6	3	5	3	1	5	6	3	5	3
45	300	2	3	5	2	3	1	2	3	5	2	3
46	300	0	1	1	1	1	1	0	1	1	1	1
47	300	6	5	3	4	4	1	6	5	3	4	4
48	300	2	1	3	2	2	1	2	1	3	2	2
49	300	9	10	8	9	13	1	9	10	8	9	13
50	300	4	2	2	2	3	1	4	2	2	2	3

Tabela C.15. Resultados de Rede de frequência de disparo de transições até 300 tiquetaques de relógio (5 filósofos com glutão)