Statecharts Estocásticos e Queuing Statecharts: Novas Abordagens para Avaliação de Desempenho Baseadas em Especificação Statecharts

Article · January 2001

CITATIONS
6

READS
78

7 authors, including:

Carlos Renato Frances
Federal University of Pará
222 PUBLICATIONS **770** CITATIONS
SEE PROFILE

Marcos José Santana
University of São Paulo
134 PUBLICATIONS **458** CITATIONS
SEE PROFILE

Solon Carvalho
National Institute for Space Research, Brazil
79 PUBLICATIONS **349** CITATIONS
SEE PROFILE

Statecharts Estocásticos e Queuing Statecharts: Novas Abordagens para Avaliação de Desempenho Baseadas em Especificação Statecharts.

Carlos Renato Lisboa Francês [1], Marcos José Santana[3],
Nandamudi Lankalapalli Vijaykumar [2], Solon Venâncio de Carvalho [2],
Regina Helena Carlucci Santana [3]

[1] Universidade da Amazônia-UNAMA
frances@icmc.sc.usp.br
[2]Laboratório de Computação e Matemática Aplicada (LAC) - INPE
{vijay, solon}@lac.inpe.br
[3]Instituto de Ciências Matemáticas e de Computação (ICMC) -USP
{mjs, rcs}@icmc.sc.usp.br

Resumo

Este trabalho investiga a utilização de uma especificação formal de alto nível, os Statecharts, com o objetivo de propiciar uma melhor compreensão de um determinado sistema, quando se pretende avaliar o seu desempenho. Além da especificação, é feita uma associação dos Statecharts com soluções analíticas e por simulação, para viabilizar o processo de modelagem como um todo. Algumas vantagens provenientes da utilização dessa nova abordagem (como hierarquia, paralelismo explícito e mecanismos de comunicação) são discutidas ao longo do artigo, assim como certas implicações da adoção dos Statecharts para avaliar desempenho. Duas abordagens formais para avaliação de desempenho são apresentadas: Statecharts Estocásticos (especificação para sistemas de filas puramente baseada em Statecharts) e Queuing Statecharts (uma aglutinação entre as representações de redes de filas e Statecharts). Um estudo de caso é apresentado ao final do trabalho para aplicar a formalização proposta.

Abstract

The paper here proposes to investigate the use of a high-level formal specification, Statecharts, to understand in a clear manner the behavior of a system where its performance evaluation is needed. The problem tackled is not only the specification but its solution by associating the specification to analytical solutions and simulation. Some of the main advantages by using this approach (such as hierarchy, explicit parallelism, and communication mechanisms) are discussed in the paper and the implications in adopting Statecharts to evaluate the performance. Two formal approaches are presented: Stochastic Statecharts (specification of queues purely based on Statecharts) and Queuing Statecharts (a merge between Queuing Networks and Statecharts). A case study is also shown to apply the proposal.

1. Introdução

O processo de modelagem com o enfoque de desempenho, na maioria de suas abordagens, não apresenta um equilíbrio entre uma boa especificação e uma solução viável para o sistema em estudo, sendo que a preocupação precípua se atem apenas ao método de solução. Essa certa "negligência" em relação à especificação pode estar associada ao perfil dos utilizadores da avaliação de desempenho, que, geralmente, possuem um conhecimento

matemático apropriado. Dessa forma, para esse tipo de usuário, um conjunto de equações de um sistema linear pode ser tão claro quanto uma especificação gráfica em alto nível.

Se, entretanto, no processo de modelagem, há outras pessoas menos especializadas envolvidas (as quais são, em algum nível, usuários do sistema a ser modelado), poderia ser interessante dispor-se de uma forma de representação clara e abstrata o bastante, a ponto de apresentar o sistema, escondendo a complexidade que pode estar associada às soluções matemáticas que alicerçam a modelagem.

Algumas das técnicas de especificação utilizadas em avaliação de desempenho, como redes de Petri, já possuem um grau de abstração e algumas características convenientes para a representação de sistemas complexos. Entretanto, este aqui examina-se a viabilidade de que características específicas providas pela notação Statecharts sejam utilizadas de maneira eficiente na avaliação de desempenho.

Com o objetivo de investigar os benefícios provenientes da especificação Statechart e sua correspondência com alguns métodos de solução, este trabalho realiza um estudo sobre as possíveis vantagens e implicações obtidas por uma abordagem baseada na adoção de uma descrição visual de alto nível, com um teor matemático associado.

A escolha dos Statecharts é justificada por algumas características adicionais, tais como paralelismo explícito, hierarquia de estados, entrada por condição ou por história e mecanismos de comunicação. Essas características associadas à visão estocástica permitem que seja feita uma análise de como está o desempenho de um determinado sistema.

2. O Processo de Modelagem

Sistemas complexos - caracterizados por uma forte demanda de atividades paralelas/concorrentes, requisitos críticos quanto aos tempos de resposta, sincronismo entre seus componentes, entre outras nuanças - são, via de regra, sensíveis a observações promovidas de maneira empírica. Paradoxalmente, esses sistemas precisam ser avaliados, de forma que se tenha uma idéia de como se apresenta o seu desempenho.

Sob esse ponto de vista, é importante que se tenha algum mecanismo que propicie uma investigação minuciosa do desempenho, sem que haja uma influência da própria aferição no resultado final da avaliação. Para esses casos, usualmente, toma-se uma abstração do sistema, denominada de modelo, a partir da qual são realizadas certas inferências, sem a necessidade de realizar-se uma efetiva experimentação. Assim, uma modelagem, no contexto de avaliação de desempenho, é um processo com um relativo teor matemático, composto por fases que são, ao mesmo tempo, independentes e harmônicas entre si. O processo de modelagem é apresentado, em suas etapas, na Figura 1.

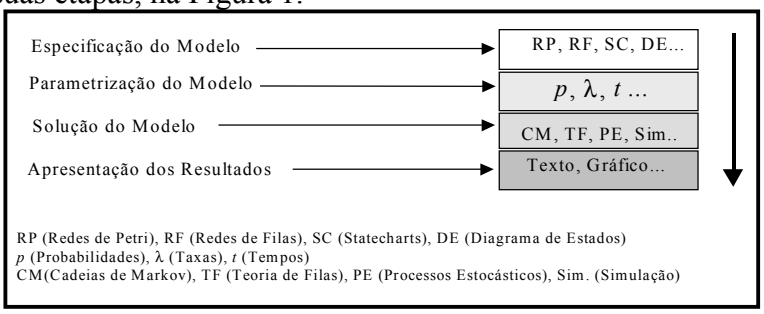


Figura 1. Etapas de um Processo de Modelagem.

De uma maneira concisa, as etapas da modelagem são descritas a seguir. Na fase inicial da modelagem, deve-se criar uma *especificação* condizente com o sistema real, na qual

devem estar contidos os componentes do sistema relevantes à avaliação, além do relacionamento entre eles. Algumas técnicas usadas para especificação são redes de filas, redes de Petri e Statecharts. Após a confecção do modelo com uma das técnicas citadas, deve-se parametrizá-lo com elementos que serão dados de entrada para a fase da solução. Na parametrização, são comuns taxas e tempos médios e probabilidades. A solução do modelo aplica um método matemático (estocástico), automatizado ou não, para adquirir medidas de desempenho a partir das entradas. Alguns métodos analíticos (cadeias de Markov, por exemplo) ou simulação são usados nessa fase. Por fim, a modelagem deve apresentar seus resultados através de uma maneira conveniente. Gráficos e arquivos-texto são geralmente utilizado nessa fase.

3. Técnicas para Especificação

Para fins de avaliação de desempenho, as técnicas de especificação devem ser baseadas em espaços de estados e eventos (interferências que causam mudanças de estados no sistema). Essa restrição é devida à relação que há com os métodos de solução, os quais são fortemente baseados em estados e suas transições. As técnicas mais referenciadas na literatura especializada são redes de filas, redes de Petri (principalmente em suas versões estocásticas) e Statecharts. Cada uma delas possui vantagens e "deficiências", quando analisadas sob a ótica do desempenho. Algumas dessas peculiaridades são discutidas a seguir.

As *redes de filas (RF)* possuem uma base matemática bastante sólida, contudo sua representação gráfica oferece apenas os elementos fila e servidor, o que em muitos casos pode ser o suficiente. Entretanto, em alguns sistemas, desejam-se representar situações que não constituem necessariamente uma fila ou um servidor. Por exemplo, um processo em um computador pode se encontrar em três possíveis situações: Processando, Bloqueado ou Pronto. Essas situações são estados abstratos que são facilmente representados em técnicas como redes de Petri e Statecharts, mas por não consistirem em filas ou servidores, não são tratadas apropriadamente em redes de filas. E mesmo filas e servidores necessitam de uma representação mais minuciosa, através da qual possam ser especificadas as várias situações que tanto fila (uma fila pode estar vazia ou não) quanto servidor (um servidor pode estar livre ou ocupado) podem se encontrar em um determinado instante.

A despeito dessas limitações, a representação do caminho linear que os clientes traçam através do sistema é descrita com bastante propriedade, noção que na maioria das técnicas é perdida com facilidade.

Redes de Petri (RP) possuem características bastante interessantes em sua representação. Uma das mais interessantes é a possibilidade da individualização de clientes, recurso que não é usual na maioria das técnicas. Há situações em que essa característica é primordial, por exemplo, na migração de processos para balanceamento de carga entre máquinas, onde se pode desejar migrar exatamente um determinado processo, e por isso deve-se saber exatamente onde ele se encontra. Entretanto, há uma certa dificuldade na representação de processos paralelos, mesmo utilizando-se da rede distribuição (rede elementar para representar criação de processos paralelos). Esse aspecto piora à medida que o modelo cresce, o que aliás é outro aspecto a ser ponderado. As redes originais, por não possuírem nenhum mecanismo de hierarquia, tendem a fazer com que os modelos cresçam substancialmente, quando a complexidade dos mesmos aumenta. Há algumas extensões que tentam minimizar esse efeito, como as RP Hierárquicas, que são baseadas em um elemento denominado superpágina [6]. O "inconveniente" dessa extensão é que as superpáginas são caixas-pretas, ou seja, são retângulos que escondem uma complexidade que,

em certos casos, é necessária para a compreensão do modelo. Além disso, algumas extensões desfiguram a notação original, o que, às vezes, parece ser outra técnica à parte, e não uma derivação das redes de Petri.

Na representação *Statecharts (SC)*, a forma explícita de algumas características (como o paralelismo e a hierarquia entre estados) é um atrativo considerável. Essa peculiaridade permite que relacionamentos complexos entre componentes de um determinado sistema sejam mostrados de maneira mais efetiva, o que não é contemplado na maioria das técnicas. Estados *default* também acrescentam a idéia de passo inicial do sistema, o que nas redes de Petri é realizado através da presença dos *tokens* (marcação inicial).

Entretanto, uma possibilidade que os statecharts não contemplam é, quando há a presença de estados ortogonais, a descrição explícita do caminho linear seguido por um determinado cliente (que é claro em RF e RP), pois pode haver vários subestados ativos em um determinado momento, o que não significa que todos os clientes (ou um cliente em particular) tenham seguido por aquela trajetória. Além disso, a individualização de clientes (como feito com a utilização dos *tokens*), fundamental em certas situações (tal como em escalonamento de processos, onde se pode decidir por migrar um processo em particular) , não é implementada em statecharts. As principais características da representação gráfica de cada técnica é sintetizada na Tabela 1.

Tabela 1. Características das Técnicas de Especificação

Técnica Característica	Redes de Fila	Redes de Petri	Statecharts
Paralelismo Explícito de Componentes	Especificado subjetivamente[1]	Especificado subjetivamente[2]	Especificado explicitamente
Mecanismos de comunicação entre componentes	Não especificado	Especificado através dos *tokens*	Especificado através dos eventos, ações e condições
Hierarquia entre os componentes	Não especificado	Não especificado, na notação original	Especificado através de superestados
Representação do caminho linear seguido por um cliente	Especificado pelas setas que unem os centros de serviço	Especificado através do fluir dos *tokens* dentro do sistema	Especificado subjetivamente através dos eventos e ações
Marcação Inicial	Não especificado	Especificado através dos *tokens*	Espeficicado através dos estados *default*
Tratamento estocástico	Espeficado	Especificado em extensões como SPN e GSPN[3]	Não especificado

As técnicas de especificação redes de filas, redes de Petri e statecharts, apresentadas superficialmente nesta seção, são abordadas com profundidade, respectivamente, em [9], [6] e [2]. Pela relevância para o trabalho, os Statecharts são apresentados com mais detalhes a seguir.

3.1. Statecharts (SC)

A abordagem sugerida nesta seção é baseada em [2], cujo teor é mais informal, com um caráter mais expositivo. Semântica e sintaxe formais são apresentadas em [3].

---

[1] Apesar de não especificado explicitamente, subentende-se que os centros de serviços possuem um funcionamento autônomo, como componentes paralelos.

[2] Apesar de haver a possibilidade de criarem-se atividades paralelas (rede "distribuição"), não há uma representação orientada a componentes (conjunto de estados) paralelos.

[3] *Stochastic Petri Nets (SPN)* são apresentadas em [7]. *Generalized Stochastic Petri Nets (GSPN)* são abordadas em [1].

Statecharts é uma extensão do diagrama tradicional de estados-transições, aos quais foram adicionadas algumas características peculiares. Os principais conceitos adicionados são hierarquia entre estados (*depth*), ortogonalidade (representação de atividades paralelas) e interdependência entre estados (mecanismos de comunicação). Além disso, statecharts são fundamentados nos seguintes elementos básicos: Estados, Eventos, Condições, Ações, Expressões, Variáveis, Rótulos e Transições. Sucintamente, cada elemento básico é apresentado a seguir.

Estados são usados para descrever componentes (e suas possíveis situações) de um determinado sistema. Os estados de um *statechart* (que representam os valores das variáveis do sistema em um determinado instante) podem ser classificados em dois grupos: básicos e não-básicos. Os *estados básicos* são aqueles que não possuem subestados. Já os *não-básicos* são decompostos em subestados. Essa decomposição pode ser de dois tipos: OR ou AND. Se a decomposição é do tipo OR, então o sistema sempre estará em um único subestado em um certo instante. Entretanto, se a decomposição é do tipo AND, o estado estará em mais de um subestado, simultaneamente.

Eventos são considerados a entidade que causa uma interferência no comportamento atual do sistema, levando esse sistema a outro comportamento. Opcionalmente, a um evento pode ser anexada uma condição (entre parênteses), também chamada de condição-guarda, de maneira que o evento só ocorrerá se satisfeita aquela determinada condição. Os statecharts proporcionam alguns eventos especiais como *true (condição)* e *false (condição)*, abreviados na notação statecharts para *tr (condição), fs (condição)*, respectivamente.

O elemento ação é considerado para representar os efeitos do paralelismo em statecharts (a influência de um estado paralelo em outro, também ortogonal). Ações podem ser uma mudança de uma expressão, uma mudança de uma variável ou eventos que são disparados em outros componentes paralelos.

As transições são a representação gráfica para denotar uma mudança de estado dentro do sistema. Rótulos podem ser acrescentados às setas para prover algum significado adicional.

4. Statecharts Estocásticos e Adaptações Necessárias

Para efeito de caracterização do problema, será especificado o funcionamento básico de um sistema de filas, através de eventos que acontecem indistintamente em um sistema de filas genérico.

Em um sistema de filas, há quatro eventos-padrão que caracterizam bem a dinâmica do sistema. O primeiro é o ato da geração dos clientes em uma determinada fonte, com um determinado ritmo. O segundo evento se constitui na chegada dos clientes à fila. O terceiro, na tomada do servidor por um determinado cliente, obedecendo a um determinado algoritmo de escalonamento. O último evento, pela saída do cliente do interior do servidor. É importante observar que todos esses eventos são tomados em um determinado instante, através de uma observação contínua do tempo. Assim, um evento fica bem caracterizado como uma função de estados de origem e de destino e do instante em que ele ocorre:

$$ev : f(s_i \rightarrow t_j), \quad \forall\, s_i \in S, t_j \in T, com\ i,\ j = 1, \mathrm{K}\ , n$$

Onde S é o conjunto discreto de estados, T é o conjunto contínuo dos tempos.

Com relação ao conjunto T, há um subconjunto dos instantes em que os eventos ocorrem, que pode ser descrito da seguinte maneira:

$$\acute{E}\hat{O} = \{\tau^{g}\} \cup \{\tau^{cf}\} \cup \{\tau^{ps}\} \cup \{\tau^{ss}\}, \quad IT \subset T.$$

Onde τ^g é o instante da geração, τ^{cf} é o instante de chegada à fila, τ^{ps} é o instante em que o cliente toma posse do servidor, τ^{ss} é o instante de saída do servidor. As diferenças entre os elementos $\tau^g_{i+1} - \tau^g_i$ e $\tau^{ss}_i - \tau^{ps}_i$ *(i=1,2,...,n)* descrevem as variáveis aleatórios *tempo entre chegadas* e *tempo de serviço*, respectivamente. As quais devem ser valores conhecidos *a priori* em qualquer avaliação de desempenho pretendida em sistemas de filas.

Pelas definições anteriores, pode-se observar que há sempre um tempo entre ocorrências de eventos, apesar de a ocorrência dos eventos, por definição, não demandar tempo, isto é, os eventos ocorrem de maneira imediata. O que há é um tempo entre as gerações e para o atendimento dos clientes, tarefas que acontecem em um determinado componente do sistema (por exemplo, um processador), o qual é representado em Statecharts (ou em algum outro diagrama orientado a estados) por um estado (ou um agrupamento deles).

Assim, da idéia de que tarefas são realizadas dentro de determinados estados, consumindo um certo tempo gera a proposição de estados que possuem tempos de permanência associados, mesmo que esse tempo seja igual a zero (caso dos estados que transicionam de maneira imediata). A notação gráfica, assumida neste trabalho, para diferenciar os estados com retardos e os estados imediatos é apresentada a seguir (Figura 2).

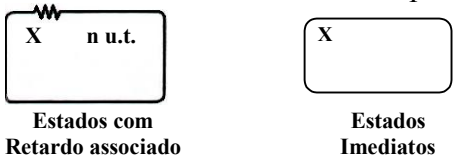


Figura 2. Estados com retardo e imediatos.

Apesar de utilizar a representação idêntica à sugerida por D. Harel [2] para estados com *delays* e *timeouts*, a semântica aqui é ligeiramente modificada, com o intuito de obter uma melhor adequação às necessidades das situações de sistemas de filas. Na semântica original, a partir da ocorrência de um evento, o estado dispara um temporizador implícito, que conta o número de unidade de tempo apresentado na inequação. Apenas após decorrido esse tempo, o sistema abandona o estado, funcionando como um *timeout* para uma determinada atividade. Já na representação sugerida neste trabalho, o sentido é que tempo ou é uma média dos tempos entre chegadas (para o caso dos estados-fonte) ou ele é um tempo médio de serviço (para os casos de servidores), ambos assumidos como exponencialmente distribuídos. Assim, o tempo que o temporizador conta, a partir da entrada no estado, é uma das variáveis aleatórias que devem ser conhecidas previamente, como comentado anteriormente.

Essa nova abordagem dada ao *delay* traz em seu cerne uma complexidade que, à primeira vista, pode não estar muito aparente: a determinação da próxima configuração, levando-se em consideração que o passo não é mais uma única unidade de tempo relacionada à execução dos Statecharts. A seção 4.2 apresenta as funções que determinam a próxima configuração a ser tomada pelo modelo, a partir da observação dos tempos nos estados não imediatos.

4.1. Escolha por Condição Probabilística

Em sistemas de filas, é necessário atribuir um valor probabilístico a cada possível caminho a ser seguido por um cliente. Essa circunstância leva à situação na qual a escolha dentre os vários caminhos possíveis está atrelada a uma probabilidade, o que introduz a idéia de uma escolha realizada através desse parâmetro. A Figura 3 apresenta uma notação semelhante àquela utilizada para entrada por condição, substituindo-se apenas a letra C pela letra P (de probabilidade).

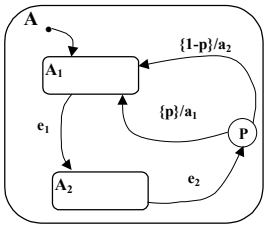


Figura 3. Escolha por probabilidade

A interpretação dada à Figura é que uma vez abandonado o estado A_2, e_2 pode estar condicionado à probabilidade *{p}* ou à *{1-p}*, e a partir do caminho escolhido, será disparada ou a ação a_1 ou a ação a_2, em algum componente paralelo não especificado na Figura.

Formalmente, a condição probabilística é definida como:

a) $\exists$ P=$\{p_1, p_2, ..., p_n\}$, onde P é o conjunto das probabilidades associadas aos eventos e p_i é cada ponto do espaço amostral definido por P$\therefore$ $p_i \in$ P, com i=1,2,...,n;
b) $0 \leq p_i \leq 1, \forall\, i = 1, 2, \ldots, n$;
c) $\sum_{i=1}^{n} pi = 1$;
d) Assim, admite-se que P $\subset$ C $\therefore$ $p_i \in$ C.

Por extensão, se $c \in$ C e $p_i \in$ C, então *ev(c)* e $ev(p_i)$ são evento atrelado a uma condição *c* e evento condicionado a uma probabilidade p_i, respectivamente. Com o intuito de diferenciar a condição probabilística das demais, usa-se a notação particular $ev\{p_i\}$, em lugar da notação original *ev(c)*, conforme sugerido em [10].

4.2. Redefinição de Passos e Configurações para Statecharts Estocásticos.

Neste ponto, volta à tona a discussão a respeito do caráter temporal dos passos e configurações, que, se neglicenciado, poderá gerar um caráter de não-determinismo entre as sucessivas configurações do sistema. Pela definição de D. Harel [3], um sistema é não-determinístico em SC se há duas reações possíveis (Υ_1, Π*) e (Υ_2, Π*), tal que $\Pi_1^* \neq \Pi_2^*$ ou $\Upsilon_1 \neq \Upsilon_2$.

Assim, uma configuração e um passo têm uma interpretação se todos os seus estados são considerados imediatos, e têm outra interpretação diante da presença de estados com retardos associados. Diante do exposto:

- Definição 1a: um *estado* é considerado *com retardo* quando existe uma variável τ_i, a qual quando avaliada determina :
  i. O tempo médio entre chegadas de clientes, caso o estado seja a fonte geradora desses clientes (*Source*). Admitindo-se que os tempos entre chegadas são exponencialmente distribuídos;
  ii. O tempo médio de serviço destinado aos clientes, caso o estado seja um servidor (em seu estado *Busy*). Admitindo-se que os tempos de serviço são exponencialmente distribuídos.
- Definição 2a: um estado é considerado imediato quando seu retardo é considerado zero (τ_i= 0).
- Definição 3a: O tempo (σi+1) da próxima configuração alcançável SC = (X, Π, θ, ξ) obedece às seguintes premissas:
  i. A variável avaliada em ξ é τ_i, onde τ_i é cada retardo associado a um estado s_i, com i=1,..,n;

ii. $\min(\tau_i)$ é a função que indica o menor dos tempos de retardo em uma determinada configuração SC_j, com j=1,..,n;

iii. O tempo gasto em cada passo é $\tau_{\Upsilon j} = \tau_{\Upsilon j-1} + \min(\tau_i)$, com j=1,..,n. O tempo $\tau_{\Upsilon j}$ é igual ao tempo final de uma configuração SC_i ou ao tempo inicial de uma configuração SC_{i+1};

iv. Se j=1, primeiro passo, então $\tau_{\Upsilon j} = \min(\tau_i)$, pois $\tau_{\Upsilon j-1} = 0$;

v. $\tau_{i.resto} = \{\tau_i\} - \min(\tau_i)$, $\forall\ \tau_i \neq \min(\tau_i)$, onde $\tau_{i.resto}$ é o restante de tempo de retardo de um estado que extrapola para o próximo passo;

vi. $\tau_{total} = \sum_{j=1}^{n} \tau_{\Upsilon j;}$

vii. $\forall\{\tau_i\} - \{\min(\tau_i)\}$, $\tau_i := \tau_{i.resto}$, a cada final de passo;

viii. Se $\tau_i = \min(\tau_i)$, então $\tau_{i.resto} = 0$ e τ_i no passo seguinte começa com valor 0.

- Definição 4a: uma escolha por probabilidade leva o sistema a reações diferentes (Υ_1, Π^*_1), (Υ_2, Π^*_2), ..., (Υ_n, Π^*_n), tal que $\Pi^*_1 \neq \Pi^*_2 \neq ... \neq \Pi^*_n$ ou $\Upsilon_1 \neq \Upsilon_2 \neq ... \neq \Upsilon_n$.
- Definição 5a: se, em um Statechart, as suas configurações e o tempo de seus passos são determinados pelas definições de 1 a 4, então esse Statecharts é Estocástico.

Pelas definições acima e, utilizando o exemplo de um servidor de arquivos, agora na notação estendida para a visão estocástica (Figura 4), com os valores hipotéticos $\tau_{source} = 5$ u.t., $\tau_{busy.proc} = 3$ u.t. e $\tau_{busy.disc} = 4$ u.t, pode-se traçar uma linha do tempo para determinar o tempo de cada passo, que são o início e o fim de cada configuração. Além disso, pode-se determinar a ordem em que as configurações ocorrem.

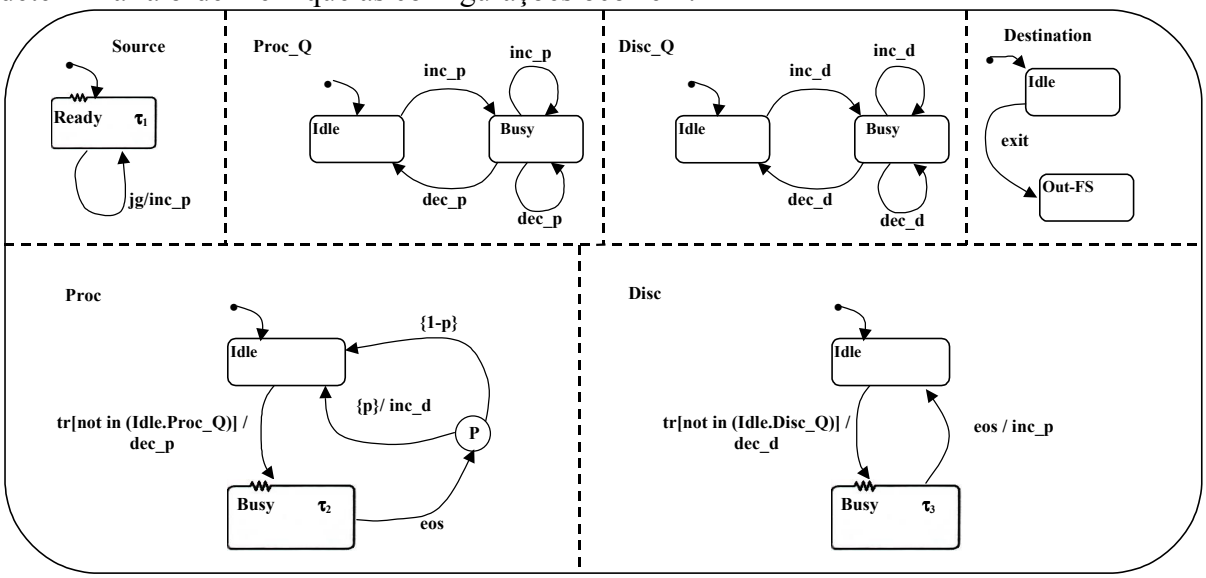


Figura 4. Servidor de Arquivos e seus Estados com Retardos Associados.

Seja SC1 a configuração dos estados *default* que apresenta o ingresso do sistema no estado *Source* (um estado com retardo de 5 u.t.), e seja SC2 a configuração sucessora, que é função da primeira, em relação ao tempo. Uma vez alcançado *Source*, o sistema "espera" por 5 u.t. até gerar o primeiro cliente (evento jg). Após decorridas as 5 u.t., o passo 1 é completado com a execução da ação inc_p (em *Proc_Q*), levando a *Busy.Proc_Q*. Na configuração seguinte, SC3 habilita de maneira imediata as transições dos eventos jg e tr[not in(Idle.Proc_Q)], e alcança simultaneamente *Source* e *Busy.Proc*. O tempo do passo e a

próxima configuração serão determinados pelas funções descritas anteriormente (de acordo com a visão estocástica), da seguinte maneira (sendo $\tau_1=\tau_{source}$, $\tau_2=\tau_{busy.proc}$, $\tau_3=\tau_{busy.disc}$) :

- Tempo do passo 1 é $\tau_{Y1} = \min_{passo1}(\tau_{source}) = 5$, pois só há um estado com retardo (*Source*);
- $\min_{passo2}(\tau_{source}, \tau_{busy.proc}) = (5, 3) = 3$;
- Tempo do passo 2 é $\tau_{Y2} = \tau_{Y1} + \min(\tau_1, \tau_2) \Rightarrow \tau_{Y2} = 5 + \min(5, 3) = 8$, ou seja, o próximo passo começa em 8 u.t.;
- O tempo que extrapola o tempo do passo 2 para os estados diferentes daquele que possui o tempo mínimo é $\tau_{i.resto} = \tau_i - \min(\tau_i) \Rightarrow \tau_{1.resto} = \tau_1 - \min(\tau_1, \tau_2) = 5 - 3 = 2$, isto é, o tempo que extrapola do passo 2 para o passo 3 é de 2 u.t. no estado S*ource*;
- Quando o passo 2 completar 8 u.t. (ao seu término), então $\forall\{\tau_i\} - \{\min(\tau_i)\}$, $\tau_i = \tau_{i.resto}$, significando que os estados que não completaram o seu retardo em um determinado passo, começam o próximo com os seus tempos iguais ao restante do passo anterior. Para o exemplo, $\tau_1 = \tau_{i.resto} = 2$. A Figura 5 apresenta a linha do tempo, com os tempos de cada passo, de acordo com os valores obtidos anteriormente.

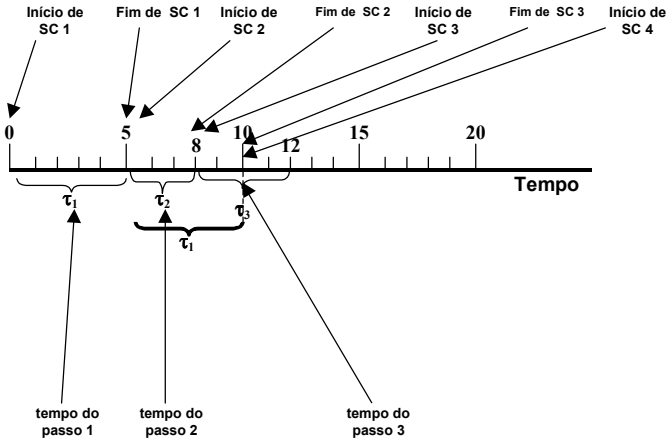


Figura 5. Linha do Tempo com suas Configurações e Passos.

Os valores temporais para SC3 são calculados como segue:

- $\min_{passo2}(\tau_{source}, \tau_{busy.disc}) = (2, 4) = 2$;
- Tempo do passo 3 é $\tau_{Y3} = \tau_{Y2} + \min(\tau_1, \tau_3) \Rightarrow \tau_{Y3} = 8 + \min(2, 4) = 10$, ou seja, o próximo passo começa em 10 u.t.;

O tempo que extrapola o tempo do passo 3 para os estados diferentes daquele que possui o tempo mínimo é $\tau_{i.resto} = \{\tau_i\} - \min(\tau_i) \Rightarrow \tau_{3.resto} = \tau_3 - \min(\tau_1, \tau_3) = 4 - 2 = 2$, isto é, o tempo que extrapola do passo 3 para o passo 4 é de 2 u.t. no estado *Busy.Disc*.

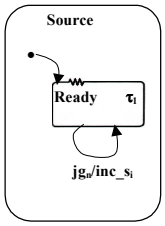
Para proceder-se se admitindo as premissas anteriores, devem-se estabelecer algumas considerações. Primeiramente, é importante esclarecer onde se encontra o caráter aleatório da especificação e das funções admitidas. A aleatoriedade é intrínseca às distribuições de geração e de atendimento ao cliente, pois são essas distribuições que geram os ritmos de chegada e de serviço (variáveis aleatórias que servem de entrada para a solução do sistema). As funções definidas como premissas garantem que os eventos possam manter a ordem esperada para um sistema de filas genérico, o que não prejudica em nada a abordagem estocástica.

Outro ponto a ser considerado é o fato de haver um certo padrão de eventos e de estados que representam os sistemas de filas, de um modo geral. Na verdade, há um padrão referente aos estados e eventos para representar sistemas de filas, apesar das possíveis variações de

nomenclatura. A próxima seção apresenta um conjunto de *templates* que visam à representação mais uniforme desses sistemas característicos.

4.3. Templates e Eventos-Padrão para Sistemas de Filas.

Esta seção se destina a apresentar um conjunto de quatro *templates* e seus eventos-padrão para especificação de sistemas de filas. No contexto deste trabalho, *template* é um conjunto de estados (possivelmente unitário) que define o funcionamento básico de um determinado componente do sistema. A idéia é que, à exceção de algum parâmetro variável (por exemplo, valores dos tempos de serviço em um servidor), o *template* seja aplicável para um determinado componente, com o mínimo possível de modificação.
Para um sistema de filas simples, são estabelecidos os seguintes *templates* (Figuras 6 a 9):

(a) 

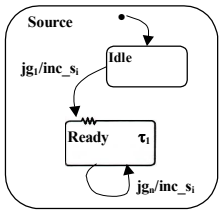
(b) 

Figura 6. *Templates* para fontes geradoras de clientes.

No *template (a)*, não há uma geração imediata do primeiro cliente, pois o estado *default* já é um estado com retardo, que uma vez alcançado, deve-se esperar transcorer o τ_i associado a ele. Já no *template (b)*, há a geração de um cliente no instante zero, em virtude do estado *default* ser um estado imediato, o que não demanda tempo. Em ambos, o evento jg (*job generation)* é responsável pela geração de um cliente, e a ação inc_s acrescenta o cliente gerado na fila de um servidor s. A escolha entre os dois depende exclusivamente das características do sistema em estudo.

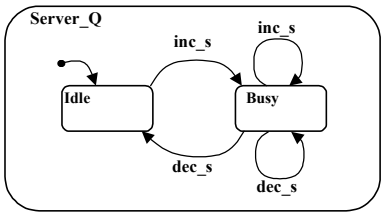


Figura 7. *Template* para filas de servidores.

O *template* fila é composto por dois subestados: Idle e Busy. Por *default*, a fila se encontra vazia (Idle), e a cada geração de cliente, há um acréscimo unitário na fila, levando o estado para *Busy*. Ocorrências reiteradas do evento inc_s mantêm a fila em *Busy*, assim como reiterações de dec_s decrementam a fila até o valor limite unitário, a partir do qual, a fila muda para Idle, indicando a ausência de clientes. Vale ressaltar que um decréscimo na fila indica que um cliente alcançou o servidor.

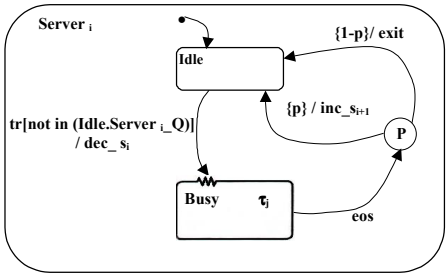


Figura 8. *Template* para Servidores.

O *template* para servidores é composto por dois subestados: Idle e Busy, com os significados de certa forma óbvios, ressaltando-se que em *Busy* há um retardo de τ_i, que significa um valor médio de uma distribuição de probabilidade. Se houver algum cliente na fila, o evento *tr[not in (Idle.Server$_i$_Q)* pode ser executado e sua transição é habilitada e a fila decrementada, em virtude de ter ido um cliente para o servidor. Após o atendimento em *Busy*, *eos* pode ser executado através de uma de suas condições probabilísticas. A ida a outro servidor implica envio de um cliente para a fila desse servidor (inc_s_{i+1}), e, caso contrário, o cliente abandona o sistema. É importante observar que pode haver mais de duas escolhas possíveis, cada uma com a sua respectiva probabilidade, assim como pode existir apenas uma escolha possível, onde se faz desnecessária a utilização da escolha por probabilidade.

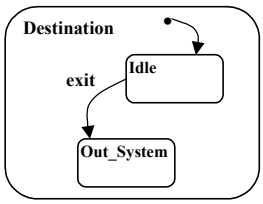


Figura 9. *Templates* para Sorvedouros.

O *template* de sorvedouro (Destination) é composto por dois subestados, que indicam que o sorvedouro está à espera de alguma resposta (Idle) ou que ele recebeu a resposta, podendo então abandonar o sistema.

De maneira a uniformizar também os eventos interessantes à avaliação de desempenho, alguns eventos são padronizados na especificação baseada nos *templates* sugeridos anteriormente, o que não impede que se criem outros eventos que aumentem a clareza da especificação, quando isso for necessário. A Tabela 2 apresenta o rótulo dos eventos, além da semântica atribuída a cada um deles.

Tabela 2. Eventos-Padrão e suas Semânticas.

Rótulo do Evento	Semântica Atribuída
jg (*Job Generation*)	Geração de um cliente obedecendo a uma determinada distribuição de probabilidade.
inc_S (*Increment of Server Queue)*	Incrementa a fila de um determinado servidor S.
dec_S (*Decrease of Server Queue*)	Decrementa a fila de um determinado servidor S.
tr[not in (Idle.Server_Q)]	Assegura que a fila do servidor não está vazia, e que próximo cliente pode ir ao servidor.
eos {p} (*End of Service*)	Indica o término do atendimento a um cliente, e uma escolha probabilística para determinar o caminho a ser seguido pelo cliente.
exit	Indica que o cliente sai do sistema que o provê de um determinado serviço e vai até um sorvedouro.

5. Queuing Statecharts

Esta seção se destina a apresentar uma alternativa a uma deficiência da especificação dos Statecharts: a dificuldade de se representar o caminho linear que um determinado cliente traça durante a sua passagem pelo sistema. Essa representação é suficientemente coberta pela especificação de redes de filas, através da ligação seqüencial estabelecida entre os componentes do modelo.

Não obstante, redes de filas não representam adequadamente a complexidade que os servidores necessitam para que suas descrições se tornem esclarecedoras, o que pode ser feito de forma bastante natural em Statecharts. Assim, para representar sistemas de filas, redes de filas e Statecharts possuem representações complementares.

Partindo-se dessa premissa, propõe-se uma aglutinação entre a especificação de redes de filas e a de Statecharts, visando a prover uma especificação mais completa para sistemas de filas.

5.1. Definição de Queuing Statecharts e seus Templates

Queuing Statecharts (QS) são definidos, através de seus componentes, conforme a seguir:

Centro de Serviço (Service Center):

- Definição 1b: o estado s' é um servidor se e somente:
  i. $\exists\ S^s \subset S$, que é o conjunto de todos os servidores do QS, tal que todo $s' \in S^s$;
  ii. $s' \in S \ \wedge\ \rho(s') = 2 = \{\text{Idle.Server, Busy.Server}\}$;
  iii. $\psi\ (s') = \text{OR}$;
  iv. $\delta\ (s') = \{\text{Idle.Server}\}$.
- Definição 2b: o estado s" é uma fila, se e somente se:
  i. $\exists\ S^q \subset S$, que é o conjunto de todas as filas do QS, tal que todo $s'' \in S^q$;
  ii. $s'' \in S \ \wedge\ \rho(s'') = \varnothing\ \therefore$ s" é um estado básico;
  iii. s" possui uma variável n_j (clientes na fila), com j=1,2,…k, que representa a variável aleatória número de clientes na fila;
  iv. n_j é definida como:
    (a) $n_j \in V$, op é uma operação algébrica, então $op(n_j) \in V$;
    (b) $n_j, v \in V, R \in \{=, >, <, \neq, \geq, \leq\}$, então $v\ R\ n_j \in C$.

  As únicas operações realizadas em n_j são: $n_j := n_j + w$, $n_j := n_j - w$, com w=1, 2,...,n.
- Definição 3b: o estado s é um centro de serviço, se e somente se:
  i. $s \in S \ \wedge\ \rho(s) = 2 = \{\text{Server\_Q, Server}\}$, onde $(S^s \cup S^q) \subset S$;
  ii. $\psi\ (s') = \text{AND}\ \therefore$ Server_Q $\perp$ Server;

  Ou seja, $\rho(s) \neq \varnothing$ e $\psi\ (s') = \text{AND}$, então trata-se de uma decomposição AND de s;
- Definição 4b: s^o é um estado fonte (Source) se e somente se:
  i. $s^o \in S \ \wedge\ \rho(s^o) = 1 = \{\text{Ready}\}$;
  ii. Ready ($\in s^o$) possui uma variável τ_i, que representa a variável aleatória tempo entre geração de clientes, obedecendo a uma distribuição de probabilidade exponencial;
- Definição 5: ^{o}s é um estado sorvedouro (Destination) se e somente se:
  i. $^o s \in S \ \wedge\ \rho(^o s) = \varnothing\ \therefore\ ^o$s é um estado básico.

Assim como nos Statecharts estocásticos, para Queuing Statecharts também há um conjunto de eventos-padrão, usado para definir o comportamento básico de um sistema de filas. Há a manutenção de todos as definições apresentadas anteriormente, tanto aquelas definidas em [3], quanto as definidas na seção 4, na qual são especificadas as diretrizes para os Statecharts Estocásticos. Desta forma, as definições de escolha por probabilidade, estados com retardos associados, além das redefinições de Passos e Configurações para Statecharts Estocásticos são aqui também assumidas.

A Figura 10 apresenta o servidor de arquivos da Figura 4, agora utilizando a notação de Queuing Statecharts.

Com a especificação de Queuing Statecharts há uma aglutinação entre características interessantes tanto de redes de filas quanto de Statecharts. Queuing Statecharts é um meio-termo entre a especificidade das redes de filas e o caráter generalista dos Statecharts.

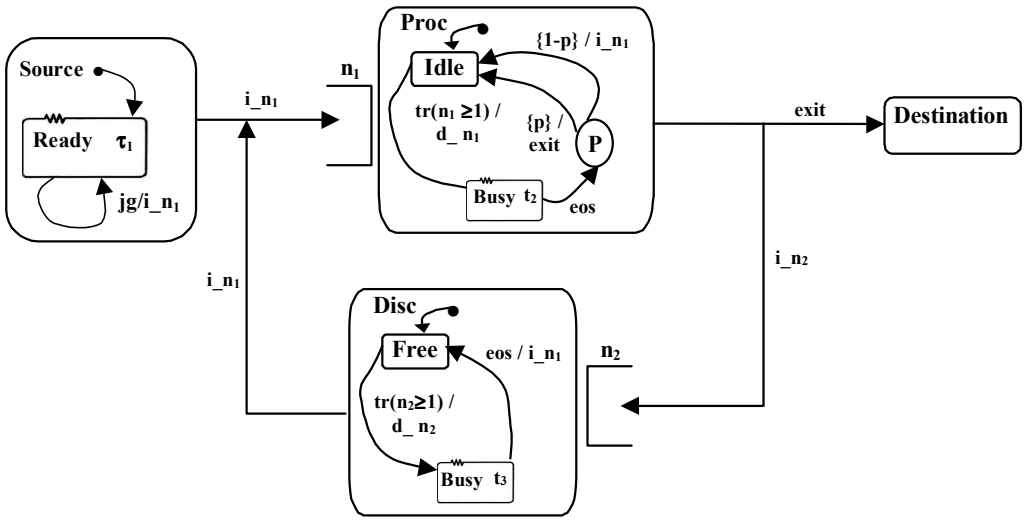


Figura 10. Servidor de Arquivos em Representação Queuing Statecharts.

5.2. Solução Analítica para Queuing Statecharts

A solução aqui adotada é baseada em redes de Jackson [4] para modelos abertos (há geração externa de clientes, assim como há saída do sistema). A solução de Jackson assume algumas restrições não discutidas aqui, mas que são perfeitamente aceitáveis para o exemplo em estudo.

Para os valores de entrada: tempo médio entre chegadas: $\tau_1 = 0{,}083$ ($\lambda = 12$); tempo médio de serviço em Proc: $\tau_2 = 0{,}04$ ($\mu_1 = 25$); tempo médio de serviço em Disc: $\tau_3 = 0{,}05$ ($\mu_2 = 20$); probabilidade $p = 0{,}6$; probabilidade $1\text{-}p = 0{,}4$.

Dessa forma, monta-se o seguinte sistema linear a partir do QS:

Para o centro 1:

$$\Lambda_1 = \underbrace{\lambda_1}_{\lambda} + \underbrace{p_{11}}_{0}\Lambda_1 + \underbrace{p_{21}}_{1}\Lambda_2 \rightarrow \Lambda_1 = \lambda + \Lambda_2$$

Para o centro 2:

$$\Lambda_2 = \underbrace{\lambda_2}_{0} + \underbrace{p_{12}}_{q}\Lambda_1 + \underbrace{p_{22}}_{0}\Lambda_2 \rightarrow \Lambda_2 = q\Lambda_1$$

Taxa de entrada no centro 1 | Taxa de chegada no centro 1 | Probabilidade de permanecer no centro 1 | Probabilidade de ir do centro 2 ao centro 1 | Taxa de entrada no centro 2

Assim, podem-se calcular as taxas de entrada para cada centro de serviço:

$$\Lambda_1 = \lambda + q\Lambda_1 = \frac{\lambda}{1-q} = \frac{12}{0{,}6} = 20 \qquad \Lambda_2 = 0{,}4 \times 20 = 8$$

Tendo-se os valores das taxas de entrada Λ_1 e Λ_2, calcula-se a utilização de cada centro:

$$\rho_i = \frac{\Lambda_i}{\mu_i} \begin{cases} \rho_1 = \frac{20}{25} = 0{,}80 \\ \rho_2 = \frac{8}{20} = 0{,}40 \end{cases}$$

Outras medidas obtidas pela solução de Jackson são:

Tabela 3. Medidas Obtidas pela Solução Analítica de Jackson.

Medida / Centro	No. médio de clientes	No. Médio de clientes na fila	Tempo de Resposta	Tempo médio de fila
Centro1	4,0	3,2	0,2	0,16
Centro2	0,66	0,26	0,08	0,03

5.3. Solução por Simulação para Statecharts Estocásticos

A solução por simulação adota a extensão à linguagem C, denominada smpl - **Si**M**ulation** **P**rogramming **L**anguage - [5]. Pelo fato do smpl ter sido proposto com uma orientação a eventos, há uma correlação direta com os eventos Statecharts dos *templates* que descrevem o funcionamento básico de um sistema de filas. Essa correlação é sintetizada na tabela 4, onde os eventos padrão dos *templates* são associados às funções smpl.

Tabela 4. Correlação entre Eventos Statecharts e Funções smpl.

Eventos Padrão da Especificação Statecharts	**Funções smpl**
jg	schedule(ev, expntl(Ta1), Customer)
inc_s	request(Servern,Customer,pri)
tr[not in(Free.Server1_Q)]	schedule(ev, expntl(Ts1), Customer)
eos	release (Servern, Customer)

{p} com um valor 60% e {1-p} para um valor 40%.	k = random(i, j);
	if ((6001 <= k) && (k <= 10000))
	schedule(ev, te, Customer);

A correlação eventos Statecharts e funções smpl é mostrada na Figura 11, na qual cada evento dos *templates* (e conseqüentemente de um sistema de filas) tem uma correspondência unívoca com as funções smpl.

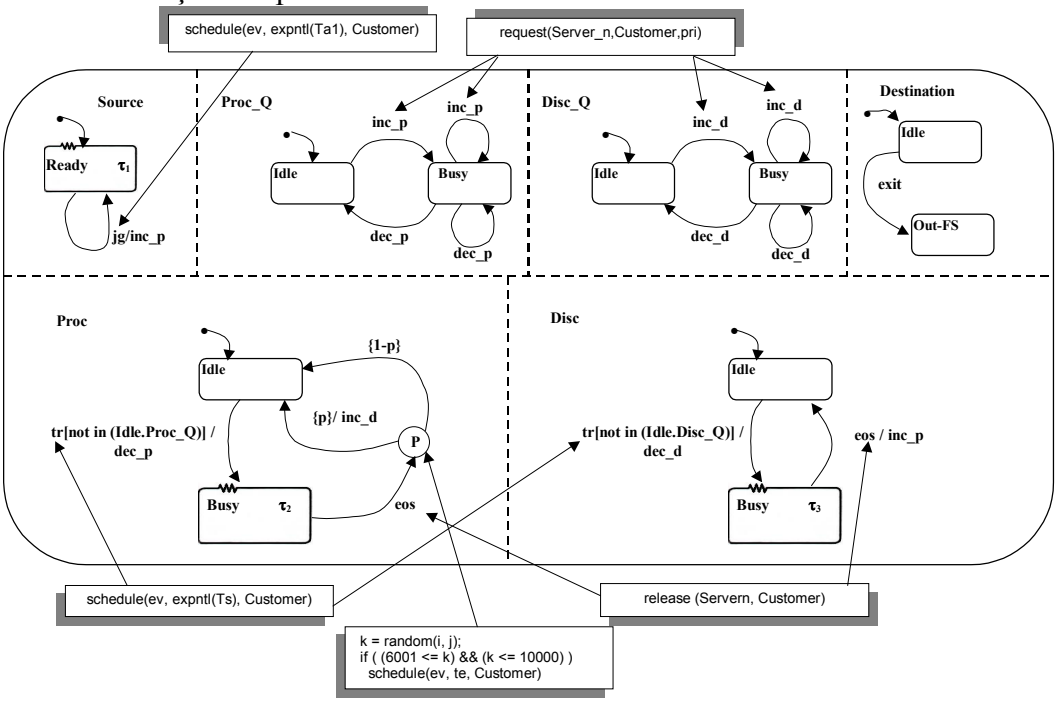


Figura 11. Associação entre Eventos Statecharts e Funções smpl.

As medidas obtidas para o exemplo, utilizando-se a solução por simulação, são apresentadas na Tabela 5, admitindo-se os mesmos parâmetros de entrada.

Tabela 5. Medidas Obtidas pela Solução por Simulação smpl.

Medida / Centro	Utilização	No. médio de clientes	Tempo de Resposta	Tempo médio de fila
Centro1	0.800	4.006	0,20	0,160
Centro2	0.400	0.667	0,083	0,033

5.4. Breve Discussão sobre as Restrições das Soluções Abordadas

A escolha entre as soluções mencionadas na seção anterior está mais atrelada a critérios subjetivos, que às restrições que cada abordagem possui. Esse assunto não é consensual, havendo duas correntes distintas de pensamento. Algumas das restrições das soluções analíticas descritas em [4] são: dificuldade no tempos de serviços não-exponenciais (serviços descritos de maneira mais realista, com distribuições como a hiperexponencial, têm um tratamento menos trivial); dificuldade na análise do estado transiente do sistema (via de regra, admite-se, na solução analítica, que o sistema se encontra em equilíbrio), dificuldade no tratamento da posse simultânea de recursos (um mesmo cliente não consegue deter mais de um recurso simultaneamente), dificuldade no tratamento de recursos passivos (como memória de computador). Para a solução por simulação, algumas das restrições listadas em [8] são: os resultados são sempre uma aproximação (por melhores que elas sejam), dependência do tratamento estatístico para os validar os resultados obtidos (cálculo do intervalo de confiança e de outras medidas estatísticas para garantir a validade dos resultados), dificuldade de lidar-se com algumas linguagens e ferramentas de simulação.

A despeito da solução adotada, as especificações apresentadas são genéricas o bastante para permitir que se utilize a abordagem de preferência do modelador.

6. Comentários Finais

Este artigo apresenta uma alternativa a algumas deficiências nas especificações mais usuais de sistemas complexos, inserindo características da representação Statecharts, tais como de hierarquização do molelo, representação explícita de componentes paralelos, mecanismos de comunicação entre esses componentes e estados *default*. A despeito do potencial dos Statecharts em representar sistemas dessa categoria, os mesmos não possuem uma abordagem voltada à avaliação de desempenho formalmente definida, apesar de já ter sido sugerido pelo seu próprio criador D. Harel, em [2], denominado à época de Markov-charts.

Para viabilizar a abordagem baseada em Statecharts foi idealizado um conjunto de *templates* para representar os componentes básicos de um sistema de filas (o ponto chave da avaliação de desempenho). A esses *templates* é sempre associado um método de solução, tanto analiticamente como por simulação. A idéia é que a especificação seja genérica a ponto de independer da solução adotada.

Dessa forma, o processo de modelagem pode contar com uma especificação mais adequada às características de sistemas complexos. Além disso, usuários menos especializados em soluções matemáticas, certas vezes não triviais, podem ter uma visão em alto nível do problema modelado.

Algumas das definições apresentadas neste artigo proporcionam um estudo passo a passo do sistema, em contrapartida à visão do estado de equilíbrio adotado em técnicas como

redes de filas. Com o objetivo de representar sistemas no estado transiente, são redefinidos os conceitos de passo e configuração, através de funções que estabelecem o tempo de duração dos passos e o algoritmo para a determinação da configuração sucessora.

Algumas outras peculiaridades não foram tratadas neste formalismo, mas estão sendo tratadas como possíveis extensões. Algumas das restrições não abordadas são geração de clientes em "lote" (diferente da geração de Poisson, apresentada neste trabalho), além de *templates* para sistemas de filas fechados (sem geração externa de clientes).

Há um interesse implícito em trazer a contribuição de potenciais usuários da especificação Statecharts para a área de avaliação de desempenho, o que otimizaria o processo de modelagem, tornando-o mais abrangente e mais bem definido.

Referências Bibliográficas

[1]CHIOLA, G., MARSAN M. A., CONTE, G. Generalized Stochastic Petri Nets: A Definition at the Net Level and Its Implications. *IEEE Transactions on Software Engineering*, vol. 19, n. 2, p. 89-106, 1993.

[2]HAREL, D. Statecharts: a Visual Formalism for Complex Systems. *Science of Computer Programming*, n. 8, p. 231-74, 1987.

[3]HAREL, D., PNUELI, A., SCHMIDT, J., SHERMAN, R. On the formal semantics of Statecharts. *IEEE Symposium on Logic*, In Computer Science, Ithaca., USA: [s.n], 1987.

[4]JAIN, R. *The Art of Computer Systems Performance Analysis – Tecnichniques for Experimental Design, Measurement, Simulation e Modeling*. s.l, John Wiley e Sons Inc, 1991.

[5]MACDOUGALL, M.H. *Simulating Computing Systems Techniques and Tools*, MIT PRESS, 1987.

[6]MACIEL, P.R.M., LINS,R.D., CUNHA, P. R. F. *Introdução às Redes de Petri e Aplicações*. Campinas, 10ª Escola de Computação, 1996.

[7]MOLLOY, M.K. Performance Evaluation Using Stochastic Petri Nets. *IEEE Trans. Comput*., v. C-31, n. 9, p. 913-17, 1982.

[8]SILVA, E.A.S., MUNTZ, R.R. *Métodos Computacionais de Solução de Cadeias de Markov: Aplicações a Sistemas de Computação e Comunicação***.** Gramado. Instituto de Informática da UFRGS, 1992.

[9]SOARES, L. F.G. *Modelagem e Simulação Discreta de Sistemas.* Rio de Janeiro, Campus Ltda, 1992.

[10]VIJAYKUMAR, N.L. *Statecharts: Their Use in Specifying and Dealing with Performance Models*. São José dos Campos, 1999. Tese (Doutorado) – Instituto Tecnológico da Aeronáutica (ITA).