UMA ESTRATÉGIA DE AVALIAÇÃO DE DESEMPENHO BASEADA EM STATECHARTS E PROCESSO MARKOVIANO DE DECISÃO

Marcelino Silva da Silva (UFPA)
marcelino@ufpa.br
Nandamudi Lankalapalli Vijaykumar (INPE)
vijay@lac.inpe.br
Carlos Renato Lisboa Francês (UFPA)
rfrances@ufpa.br
Solon Venâncio de Carvalho (INPE)
solon@lac.inpe.br

Este trabalho propõe uma estratégia que associa uma especificação formal de alto nível, Statecharts, a um Processo Markoviano de Decisão. O Statecharts é utilizado para propiciar uma representação clara do comportamento do sistema, incluinddo as possíveis tomadas de decisões e os custos advindos de cada decisão. Enquanto que o Processo Markoviano de Decisão analisa os efeitos a longo prazo de cada decisão. Como resultado dessa associação, permite-se que usuários do sistema, os quais não possuem um conhecimento profundo sobre o processo de avaliação de desempenho, possam participar de forma mais efetiva na etapa de modelagem, provendo informações que tornam a modelo mais coerente com o sistema real.

Palavras-chaves: Statecharts, Processo Markoviano de Decisão, Avaliação de Desempenho

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, via de regra, 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, por exemplo, 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.

Baseando-se nesse intuito de prover uma especificação clara e abstrata do sistema, Harel (1987) apresenta a técnica de especificação formal denominada Statecharts. Essa técnica é apresentada como uma extensão dos diagramas convencionais de estados e transições, na qual a principal característica dessa especificação situa-se na possibilidade de representar de forma clara e objetiva os vários componentes paralelos do sistema.

Para que os Statecharts possam ser utilizados no processo de avaliação de desempenho de sistemas, Vijaykumar (1999) apresenta um método computacional para que, a partir da especificação em Statecharts do sistema, possa-se obter a Cadeia de Markov a Tempo Contínuo isomórfica a essa especificação.

Dando continuidade ao trabalho apresentado de Vijaykumar (1999), este artigo propõe uma estratégia que associa uma especificação em alto nível baseada em Statecharts a um Processo Markoviano de Decisão (PMD). A partir da especificação Statecharts podem ser estabelecidas indicações de quando o sistema deve optar por uma ou por outra decisão, assim como os custos de cada decisão; possibilitando, desta forma, a construção de um Processo Markoviano de Decisão a Tempo Contínuo. De posse do PMD, pode-se obter a política ótima R^*, que orienta as tomadas de decisões ao longo do tempo, para minimizar o custo médio, em longo prazo, do sistema.

Os PMD têm sido amplamente utilizados nas mais diversas áreas do conhecimento, por exemplo, Pandana (2004) apresenta uma estratégia de transmissão de dados em uma comunicação ponto a ponto sem fio, na qual a modulação e a potência de transmissão do sinal são dadas por uma política ótima obtida por um PMD; Ghahnavie (2006) aplica PMD para a gerência de manutenção de unidades geradoras de energia em sistemas de potência; e Narasimha (2007) apresenta um sistema de gerência de e-mails baseado em uma política obtida por um PMD, com a finalidade de minimizar o tempo de trabalho do usuário gasto para ler e responder seus e-mails.

Dado a grande aplicabilidade do PMD várias ferramentas foram desenvolvidas para auxiliar a modelagem e a resolução do modelo. Sarmiento (2006) desenvolveu um framework para modelagem de PMD baseado em programação orientada a objeto, o qual permite que a partir da implementação do modelo do sistema o analista possa escolher entre distintas técnicas de soluções. Essa ferramenta torna mais rápido o processo de resolução do modelo,

entretanto ainda não permite que o sistema seja representado por uma especificação de alto nível como por exemplo Statecharts. Beccuti (2007) associa a especificação formal de alto nível Redes de Petri com PMD. Contudo, Redes de Petri apresentam certa dificuldade na representação de paralelismo e sincronização entre componentes, o que pode ser claramente representado em Statecharts. Desta forma, o trabalho demonstrado neste artigo vem como um complemento aos trabalhos apresentados anteriormente.

A especificação formal Statecharts é apresentada de forma sucinta na seção 2. Na seção 3, demonstra-se o algoritmo para construção do PMD a partir da especificação Statecharts. Resultados numéricos do exemplo selecionado são apresentados na seção 4. Por fim, na seção 5 são apresentados os comentários finais e conclusões a respeito do trabalho.

2. Especificação Statecharts

Sistemas reativos são aqueles que respondem a algum estímulo. Sistemas complexos são sistemas reativos envolvendo paralelismo, sincronização e interdependência de seus componentes ou subsistemas. Uma variada gama de aplicações pode ser classificada como tal: automóveis, controladores industriais, interface homem-máquina, protocolos de redes de comunicações, etc.

Um dos principais problemas desse tipo de sistema consiste na dificuldade de descrever o seu comportamento de forma clara, concisa e livre de ambigüidades, uma vez que esse comportamento é dirigido por eventos complexos e inter-relacionados. Tal descrição torna-se inviável por meio de diagramas tradicionais de máquinas de estado, visto que pequenas variações no comportamento do sistema podem acarretar em grandes mudanças na representação do sistema e no crescimento exponencial do espaço de estados [Harel (1987)].

Para permitir uma representação clara de hierarquia, concorrência e interdependência entre componentes do sistema, Harel (1987) apresentou uma técnica de especificação formal denominada Statecharts. Este método é uma extensão do diagrama tradicional de estados/transições, ao qual foram adicionadas algumas características peculiares.

Os principais conceitos adicionados ao Statecharts são: hierarquia de estados (profundidade), ortogonalidade (representação de atividades paralelas) e interdependência entre estados (mecanismos de comunicação). Os elementos fundamentais para se especificar um sistema reativo em Statecharts são: Estados, Eventos, Condições, Ações, Expressões, Variáveis, Rótulos e Transições. Cada um desses elementos básicos é apresentado a seguir.

Estados são usados para descreverem componentes (e suas possíveis situações) de um determinado sistema. Os estados de um Statecharts podem ser classificados em dois grupos: básicos e não-básicos. Os estados básicos são aqueles que não possuem sub-estados. Já os não-básicos são decompostos em sub-estados. Essa decomposição pode ser de dois tipos: XOR ou AND. Se a decomposição é do tipo XOR, então esse estado do sistema não poderá estar em mais de um sub-estado simultaneamente. Entretanto, se a decomposição é do tipo AND, esse estado poderá encontrar-se em mais de um sub-estado simultaneamente.

Eventos são elementos de grande importância na especificação Statecharts, pois são esses elementos que interferem no estado atual do sistema, levando o sistema a outro estado, ditando, desta forma, a dinâmica do sistema. Opcionalmente, a um evento pode ser anexada uma condição (entre colchetes "*[]*"), também chamada de condição-guarda, de maneira que o evento só ocorrerá (ou só irá interferir no sistema) se satisfizer esta determinada condição. Os

Statecharts classificam os eventos em externos e internos. Os eventos externos são aqueles que devem ser explicitamente estimulados, enquanto os internos são detectados automaticamente pelo sistema e são imediatamente estimulados sem a necessidade de um estímulo explícito.

Para se utilizar os Statecharts como ferramenta de análise de desempenho, os eventos externos (explicitamente estimulados) carregam uma informação estocástica com alguma distribuição de probabilidade, por exemplo, taxas de chegada, taxas de serviço, etc. Os internos não terão esta informação estocástica associada, ou seja, o tempo de espera para a ocorrência da transição é zero, quando detectados automaticamente. Os Statecharts proporcionam alguns eventos internos especiais como *true (condição)* e *false (condição)*, abreviados na notação Statecharts para *tr (condição)*, *fs (condição)*, respectivamente. Para tratar o sistema como um PMDTC, faz-se necessário que as taxas de ocorrência dos eventos externos sejam dadas por uma distribuição de Poisson.

O elemento ação é utilizado para descrever os efeitos do paralelismo em Statecharts. Ações podem ser uma mudança de uma expressão, uma mudança de uma variável. Para adaptar os Statecharts a serem aplicados na avaliação de desempenho, ações podem ser eventos internos.

As transições são a representação gráfica para denotar uma mudança de estado dentro do sistema. Rótulos nos arcos de transição consistem da seguinte notação: *evento[condição]/ação*. A interpretação desta notação é: quando o evento for habilitado e a condição for satisfeita, a transição é realizada. Após a transição a ação é executada. Quando a ação for um evento interno, há transição, em geral, em outros componentes paralelos.

Para ilustrar as definições apresentadas, observa-se o sistema modelado em Statecharts apresentado na Figura 1. Considera-se um sistema que possui duas máquinas idênticas atendendo a uma mesma fila de requisições. A cada instante de tempo estas duas máquinas podem ser ligadas ou desligadas através dos eventos *lig1* e *desl1*, para a máquina M1, e *lig2* e *desl2* para a máquina M2, conforme a "vontade" do controlador.

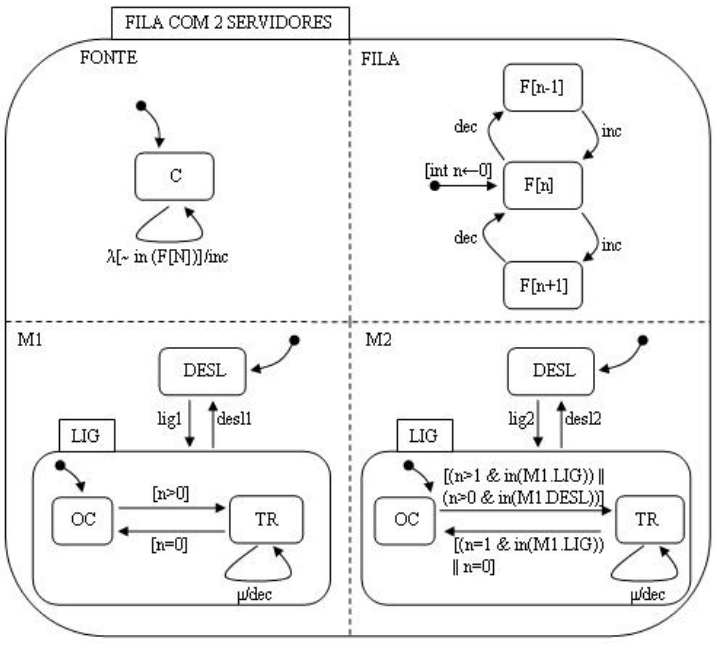

Figura 1. Modelo em Statecharts de um sistema com dois servidores e uma fila

Nesse sistema identificamos quatro componentes básicos que atuam em paralelo: FONTE (componente que representa a chegada de requisições ao sistema), FILA (representa o número de clientes no sistema, tanto os sendo atendidos quanto os em espera), M1 e M2 (representam as duas máquinas do sistema). Desta forma, o estado *raiz* do sistema pode ser modelado como um estado não-básico do tipo AND, no qual FONTE, FILA, M1 e M2 são sub-estados desse estado *raiz*. A representação gráfica de paralelismo é feita separando-se os estados paralelos por uma linha pontilhada. Cada um desses sub-estados pode ser modelado como segue.

a) **FONTE**: possui um sub-estado básico C (chegada de requisições), o qual possui uma transição que é executada quando o evento estocástico chegada de requisições ocorre, representado pelo rótulo λ. Esse evento interfere no sistema através de uma ação que dispara o evento interno *inc* no estado FILA, sob a condição de que a FILA não esteja totalmente cheia (*[not in(F(N)])/inc*);

b) **FILA**: possui N sub-estados básicos F(n), no qual cada sub-estado representa o número *n* de clientes no sistema. As transições ocorrem através dos eventos internos *inc* (o sistema muda do estado F(n) para F(n+1)) e *dec* (o estado muda de F(n) para F(n-1));

c) **M1**: possui dois sub-estados, um básico, DESL (máquina M1 desligada), e outro não-básico, LIG (máquina M1 ligada). As transições entre esses dois sub-estados ocorrem através dos eventos internos *lig1* e *desl1*, como mencionando anteriormente. O sub-estado não básico LIG é composto por dois sub-estados básicos, OC (M1 ociosa) e TR (M1 trabalhando), no qual M1 passa de OC para TR se houver alguma requisição na fila (*tr[n>0]*) e transiciona de TR para OC quando a fila estiver vazia (*tr[n=0]*). Estando no estado TR, M1 atende as requisições com uma taxa μ, interferindo no sistema através da ação *dec*, a qual dispara o evento interno *dec* no estado FILA;

d) **M2**: esse estado é decomposto conforme M1, sendo que a transição de OC para TR ocorre quando existe mais de uma requisição na FILA e a máquina M1 está no estado LIG ou quando M1 está desligada e existe pelo menos uma requisição na fila (*tr[(n>1 & in(M1.LIG)) || (n>0 & in(M1.DESL))]*) e M2 muda para OC quando existe apenas uma requisição na fila e M1 está no estado LIG ou quando não existe requisição alguma na fila (*tr[(n=1 & in(M1.LIG)) || n=0]*). Esta diferença ocorre para evitar que quando houver apenas uma requisição na fila, M1 e M2 atendam simultaneamente esta mesma requisição.

Para evitar ambigüidades no texto, o termo **estado** (o qual é utilizado em PMD para denotar a condição em que o sistema se encontra como um todo e em Statecharts possui uma utilização mais ampla, referindo-se não apenas a condição em que o sistema encontra-se, mas também a componentes específicos do sistema) será utilizado quando se estiver referindo ao estado de um Statecharts e para os PMD substituiremos o termo **estado** por **configuração** (por exemplo, "espaço de estados *S*" será referenciado como "espaço de configurações *S*")**.**

Da mesma forma, o termo **ação**, o qual em PMD indica qual decisão foi tomada e em Statecharts indica como os estados paralelos comunicam-se, continuará sendo utilizado como especificado em Statecharts, enquanto que para os PMD será utilizado sempre o termo **decisão** (por exemplo, substituiremos "a ação *a* foi tomada no estado *i*" por "a decisão *a* foi tomada no estado *i*").

3. Construção de um PMDTC a partir de um Statecharts

Nessa seção são apresentados uma extensão ao Statecharts para que este possa ser convertido em um PMDTC e o método computacional para realizar a conversão.

3.1. Adaptações Necessárias ao Statecharts

Para se obter um PMDTC a partir de um Statecharts, faz-se necessário a inclusão de um novo elemento à especificação Statecharts. Este novo elemento, denominado *estado de decisão*, indica que quando o sistema encontra-se neste estado, deve-se escolher uma e apenas uma decisão a ser tomada, a qual leva, de forma instantânea, o sistema para um novo estado. Como notação gráfica para esse novo elemento, sugere-se a utilização de um círculo com a letra D (decisão) no centro. A Figura 2 apresenta um exemplo dessa notação. Quando o sistema está no estado A, um evento estocástico é disparado com taxa λ, o qual leva o sistema a um estado de decisão. Nesse estado de decisão, deve-se optar por tomar a decisão <0>, que dispara a ação x, ou tomar a decisão <1>, que dispara a ação y.

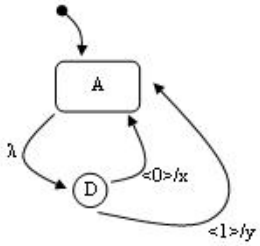

Figura 2. Exemplo de utilização do estado de decisão

No exemplo apresentado na Figura 3 o sistema, no componente CONTROLE, pode optar por três decisões: decisão <0>, a qual desliga as duas máquinas; decisão <1>, que liga apenas a máquina M1; e decisão <2>, que liga as duas máquinas. Observa-se que não existe a opção de desligar a M1 e ligar M2, isto ocorre porque, como se considerou que as duas máquinas são idênticas com os mesmos custos e taxas de atendimento, tomar a decisão de desligar M1 e ligar M2 seria, para o cálculo do custo e obtenção da política R^*, redundante com decidir por ligar M1 e desligar M2.

Para a descrição dos custos acarretados ao sistema, sugere-se a utilização de uma tabela de custos em conjunto com a especificação em Statecharts. A escolha por utilizar uma tabela para especificar os custos, em detrimento de incluir tais custos diretamente na especificação Statecharts, foi feita para evitar "sobrecarregar" a especificação Statecharts, mantendo-se, assim, a especificação clara e objetiva, como foi proposto inicialmente por Harel (1987).

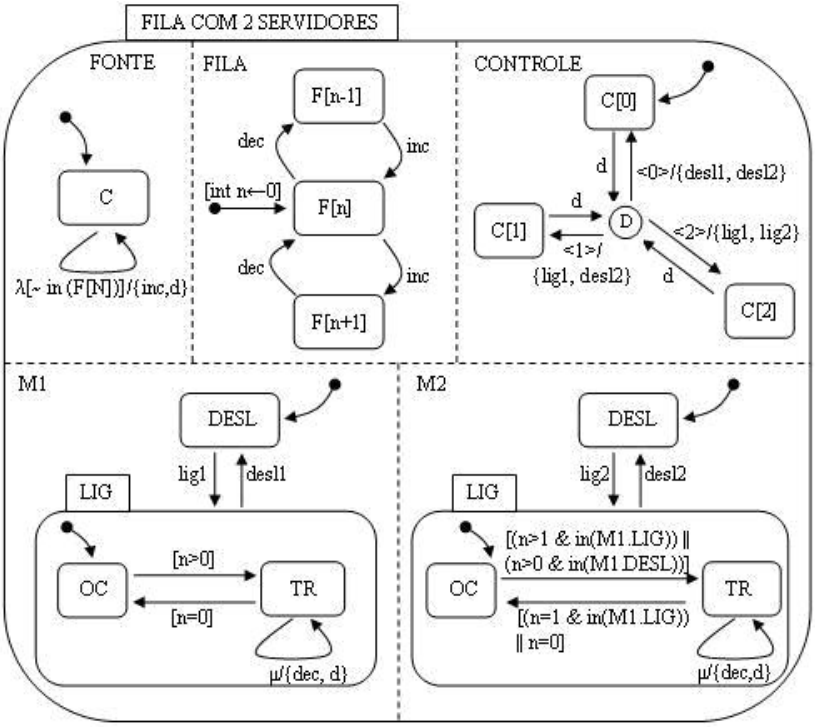

Figura 3. Modelo em Statecharts de um sistema com dois servidores e uma fila utilizando um estado paralelo para controlar as máquinas

Para montar a tabela de custos sugere-se dividir os custos em três tipos básicos: (1) custos de permanência; (2) custos de transições; (3) custos particulares.. O primeiro tipo de custo refere-se a custos que são incididos ao custo total devido à permanência do sistema em um determinado estado. O segundo tipo trata dos custos relacionados às transições entre estados. Toda vez que um evento estocástico ocorrer, uma decisão será tomada e o sistema mudará para um determinado estado e essa transição acarretará um determinado custo ao custo total. Estes primeiros dois tipos de custos possuem campos específicos na tabela por serem muito comuns em PMD. Entretanto, em alguns casos particulares, o custo de cada decisão é calculado como uma função de diversos fatores, sendo necessário que o analista o especifique de forma explicita. Nesse caso, utiliza-se o campo da tabela dedicado aos custos particulares. A Tabela 1 demonstra a tabela de custos para o sistema da Figura 3.

Tabela de custos para o sistema com uma fila e dois servidores		
Custos de Permanência		
Estado	**Custo**	
M1.LIG	cm_1l	
M2.LIG	cm_2l	
F(n)	n.cr	
Custos de Transições		
Estado Sucessor	**Estado Destino**	**Custo**
M1.DESL	M1.LIG	clm_1
M2.DESL	M2.LIG	clm_2
M1.LIG	M1.DESL	cdm_1

M2.LIG	M2.DESL	cdm_2
Custos Particulares		
$-gf(tempo(\text{M1.TR})\mu + tempo(\text{M2.TR})\mu)$		

Tabela 1. Tabela de custos para o sistema com uma fila e dois servidores

Pela Tabela 1 identifica-se que o custo total é composto por oito componentes:

a) Custo de permanecer com a máquina M1 ligada: quando o sistema estiver no estado M1.LIG, será adicionado ao custo total um custo $cm_1l.t$, onde cm_1l é o custo por unidade de tempo de manter a máquina M1 ligada e t é tempo de permanência nesse estado;

b) Custo de permanecer com a máquina M2 ligada: semelhante a manter M1 ligada;

c) Custo para manter cada requisição no sistema: toda vez que existirem n requisições no sistema (FILA está no estado F(n)) será adicionado ao custo total um custo $n.cr.t$, onde cr é o custo para manter uma requisição no sistema por unidade de tempo.

d) Custo de ligar a máquina M1: quando o estado M1.DESL fizer parte da configuração atual do sistema (configuração i), o sistema poderá passar para a configuração j em que o estado M1.LIG faz parte dessa configuração com probabilidade $p_{ij}(a)$, dado que a é a decisão tomada na configuração i. Então o custo acarretado por ligar a máquina M1 será calculado como $clm_1 \sum_{j \in s} p_{ij}(a)$;

e) Custo de ligar a máquina M2: semelhante ao custo de ligar M1;

f) Custo de desligar a máquina M1: semelhante ao custo de ligar, considerando-se que o estado muda de M1.LIG para M1.DESL;

g) Custo de desligar a máquina M2: semelhante ao custo de desligar M1;

h) Ganho por finalizar uma requisição: quando uma requisição é finalizada, um ganho $-gf$ é incidido ao sistema. Dessa forma pode-se calcular o ganho total por finalizar as requisições por unidade de tempo como $-gf(tempo(\text{M1.TR})\mu + tempo(\text{M2.TR})\mu)$. Observa-se o sinal negativo, isto indica que esse valor não é um custo e sim um ganho;

Tendo-se especificado o sistema em Statecharts com a extensão proposta e a tabela de custos, é então possível construir um PMDTC para encontrar a política ótima R^* do sistema. O algoritmo para construção do PMDTC é apresentado na seção seguinte.

3.2. Algoritmo para construção do PMDTC

Este trabalho propõe que a partir de uma especificação em Statecharts com as devidas indicações de quando o sistema deve optar por uma ou outra decisão e os custos acarretados ao sistema quando uma decisão é escolhida, é possível construir um PMDTC para obter-se a política ótima R^* que minimiza o custo médio em longo prazo do sistema.

Visto que na literatura já existem vários métodos eficientes para a obtenção da política R^*, de forma particular nesse trabalho foi utilizado o Algoritmo de Iteração de Valores, o problema, então, restringe-se a encontrar: (i) as configurações adequadas do sistema que indicam as configurações do PMDTC; (ii) as ações que podem ser tomadas em cada estado do PMDTC; e (iii) as taxas de saída de cada configuração quando certa ação foi escolhida.

Para montar um PMDTC a partir de um Statecharts, primeiramente monta-se um grafo, no qual cada nó desse grafo representa uma configuração do sistema. Admite-se que cada configuração (nó do grafo) pode ser de dois tipos: (1) uma configuração na qual o sistema não se encontra em um estado de decisão; e (2) uma configuração onde o sistema encontra-se em um estado de decisão. Cada aresta do grafo será direcionada, representada por uma seta, no qual cada seta pode ser também de dois tipos: (1) uma seta estocástica, que indica que uma transição é feita quando um evento estocástico ocorre, isto é, o tempo de espera para que essa transição seja disparada é exponencial distribuído com média $1/\lambda$, onde λ é o rótulo da seta; e (2) uma seta de decisão que indica qual deve ser o nó destino do grafo quando a decisão $<a>$ é tomada, onde $<a>$ é o rótulo dessa seta (nesse caso a transição é executada de forma instantânea).

O grafo será montado segundo a condição de que cada nó do grafo representará uma configuração do sistema, no qual a saída dessa configuração somente pode ser feita ou por uma tomada de decisão ou pela ocorrência de um evento estocástico. Essa condição implica em que as configurações em que existem eventos internos habilitados não farão parte do grafo. Isso ocorre porque nesses casos o tempo de permanência nessas configurações é zero, visto que, ao encontrar-se nessas configurações, automaticamente as transições que denotam os eventos internos habilitados serão disparadas.

A partir dessa condição, utiliza-se o seguinte algoritmo para montar o grafo:

geraGrafo(Statecharts ***sc***, Grafo ***g***)
Início
  Obtém-se a configuração inicial ***c*** de ***sc***
  Estando na configuração ***c***, se existirem eventos internos habilitados, dispara-se essas transições e continua-se reagindo até encontrar uma configuração em que não existem mais eventos internos habilitados
  Adiciona-se essa nova configuração ao grafo ***g*** e marca-se esse nó como "não expandido"
  Enquanto existir um nó em ***g*** marcado como "não expandido", executa-se o seguinte laço
  Início
    fonte é o nó marcado como "não expandido"
    Se ***fonte*** for uma configuração de decisão então
    Início
      Monta-se o conjunto ***ds***, formado pela combinação das decisões de cada estado de decisão da configuração ***fonte***
      Para cada combinação $d \in ds$ faça
      Início
        Encontra-se a configuração ***destino*** alcançada quando a combinação de decisões ***d*** é disparada a partir da configuração ***fonte***
        Estando na configuração ***destino***, se existirem eventos internos habilitados, dispara-se essas transições e continua-se reagindo até encontrar uma configuração em que não existem mais eventos internos habilitados e atribui-se essa nova configuração a ***destino***
        Se ***destino*** ainda não fizer parte do conjunto de nós de ***g***, adiciona-se ***destino*** a ***g*** e marca-se ***destino*** como "não expandido"
        Adiciona-se em ***g*** uma seta de decisão saindo de ***fonte*** para ***destino*** com o rótulo "***d***"

Fim
Fim
Se ***fonte*** NÃO for uma configuração de decisão então
Início
Monta-se o conjunto ***es*** de todos os eventos estocásticos que podem ocorrer enquanto o estiver na configuração ***fonte***
Para cada evento estocástico ***e*** ∈ ***es*** faça
Iníco
Encontra-se a configuração ***destino*** alcançada quando o evento ***e*** é disparada a partir da configuração ***fonte***
Estando na configuração ***destino***, se existirem eventos internos habilitados, dispara-se essas transições e continua-se reagindo até encontrar uma configuração em que não existem mais eventos internos habilitados e atribui-se essa nova configuração a ***destino***
Se ***destino*** ainda não fizer parte do conjunto de nós de ***g***, adiciona-se ***destino*** a ***g*** e marca-se ***destino*** como "não expandido"
Se já existir uma seta estocástica em ***g*** saindo de ***fonte*** e chegando a ***destino***, então adiciona-se "***+e***" ao rótulo da seta, senão adiciona-se a ***g*** uma seta estocástica saindo de ***fonte*** e chegando a ***destino*** com rotulo "***e***"
Fim
Fim
Marca-se ***fonte*** como "expandido"
Fim
Fim

A Figura 4 demonstra o grafo gerado para o sistema especificado no Statecharts da Figura 3. Cada nó do grafo representa uma configuração do sistema, no qual a seqüência (A B, C, D) na indicação do nó representa o estado dos componentes FILA, CONTROLE, M1 e M2, respectivamente. Como FONTE possui apenas um estado, a sua representação foi omitida no grafo. As setas pontilhadas representam setas de decisão e as setas contínuas representam setas estocásticas. Considerou-se o tamanho máximo da fila igual a três ($N=3$).

Nesse primeiro trabalho está se admitindo que toda vez que um evento estocástico ocorre, é necessário que uma decisão seja tomada. Para o exemplo da Figura 3, os eventos estocásticos são chegada de clientes, λ, e término de atendimento, μ. Sendo assim, toda vez que um desses eventos ocorre, uma decisão deve ser tomada.

Ao se observar o grafo da Figura 4, pode-se notar que, para esse caso particular onde todo evento estocástico leva a uma decisão, toda seta estocástica tem como nó fonte uma configuração sem estados de decisão e como nó destino uma configuração de decisão, enquanto que toda seta de decisão possui como nó fonte uma configuração de decisão e como nó destino uma configuração sem estados de decisão.

O espaço de configurações do PMDTC será formado pelo conjunto de configurações do sistema, no qual pelo menos um dos estados é um estado de decisão. Isto indica que cada configuração do PMDTC representa a configuração que o sistema encontra-se após a ocorrência de um evento estocástico e antes de uma decisão ter sido tomada. As possíveis decisões de cada configuração serão dadas pelas setas de decisões que saem do nó que representa cada configuração.

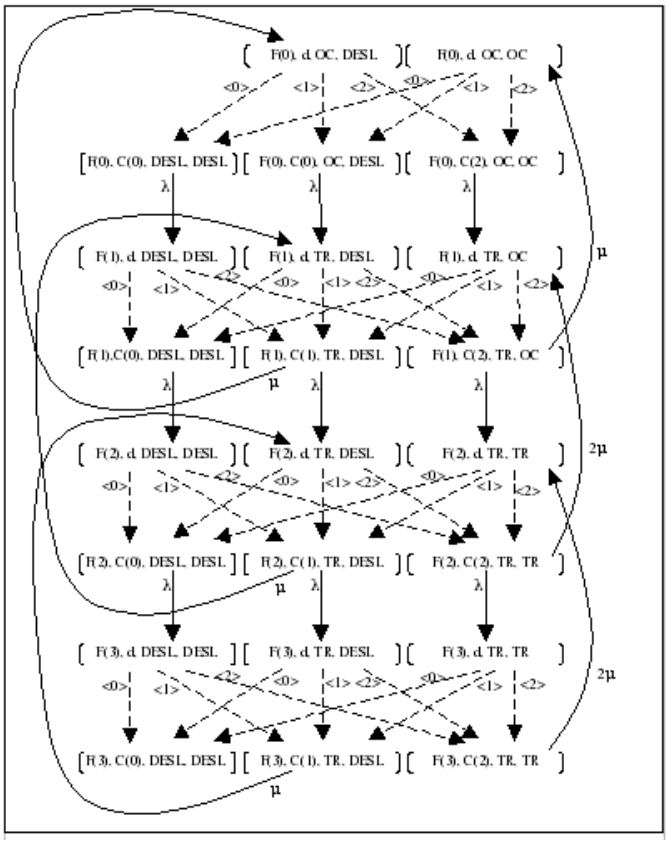

Figura 4. Grafo do sistema com dois servidores e uma fila

Para se encontrar as taxas de saída de uma configuração do PMDTC dada uma certa decisão, observa-se o nó que representa essa configuração, a decisão tomada e o nó alcançado por essa decisão. Esse último nó representa a configuração em que o sistema irá "de fato" atuar, ou seja, se o sistema encontra-se na configuração representada pelo nó ***fonte*** e a decisão ***d*** é escolhida, então o sistema irá operar na configuração representada pelo nó ***auxconf***. Dessa forma, as taxas de saída da configuração representada pelo nó ***fonte***, quando a decisão ***d*** é tomada, serão indicadas pelas setas estocásticas que saem de ***auxconf***. Da mesma forma, cada seta estocástica saindo de ***auxconf*** irá indicar qual configuração é alcançada quando o sistema encontra-se na configuração ***fonte*** e a decisão ***d*** é escolhida.

O algoritmo que executa esse processo de conversão é mostrado a seguir:

geraPMDTC(Grafo ***g***, PMDTC ***pmd***)
Início
Para cada nó do grafo ***g*** que representa uma configuração de decisão faça
Início
fonte é o nó marcado representando a configuração de decisão
Monta-se o conjunto ***ds***, formado pelas setas de decisões saindo do nó ***fonte***
Para cada seta de decisão ***d*** ∈ ***ds*** faça
Início
Encontra-se o nó ***auxconf*** alcançado pela seta ***d*** a partir do nó ***fonte***

Monta-se o conjunto ***es*** de todas as setas estocásticas saindo do nó ***fonte***
Para cada seta estocástica ***e*** ∈ ***es*** faça
Início
Encontra-se o nó ***destino*** alcançado pela seta ***e*** a partir do nó ***auxconf***
Se a configuração representada pelo nó ***fonte*** ainda não faz parte do espaço de configuração do ***pmd***, então se adiciona ***fonte*** ao ***pmd***
Se a configuração representada pelo nó ***destino*** ainda não faz parte do espaço de configuração do ***pmd***, então se adiciona ***destino*** ao ***pmd***
Adiciona-se ao ***pmd*** a decisão ***d*** para a configuração ***fonte*** e adiciona-se que se a decisão ***d*** é tomada na configuração ***fonte***, o sistema muda para o estado ***destino*** com taxa ***e***
Fim
Fim
Fim
Fim

Para obter a política ótima R^* que minimiza o custo médio a longo prazo do PMDTC obtido pelo algoritmo demonstrado, encontra-se os custo de cada configuração dado que a decisão *a* foi escolhida a partir da tabela de custos e utiliza-se o AIV.

Após executar o AIV obtêm-se os resultados gerais do sistema: (1) política ótima que indica qual decisão deve ser tomada quando o sistema se encontra na configuração *c*; (2) custo médio total a longo prazo e custos individuais como descriminado na tabela de custos; e (3) probabilidades (em regime estacionário) do sistema estar em cada uma das configurações.

4. Resultados Numéricos

Para verificar a conformidade e confiabilidade do PMDTC gerado pelo método computacional apresentado neste trabalho, utilizou-se o sistema demonstrado na Figura 3, sendo que se aumentou o número de máquinas para dez ($K=10$) e o tamanho da fila para cem ($N=100$). Este mesmo sistema já foi modelado e analisado por Tijms (1994), o qual utilizou de artifícios matemáticos para diminuir o espaço de estados do PMDTC. Desta forma, os resultados gerados pelo método apresentado nesse artigo são comparados com os obtidos por Tijms (1994) para se verificar a confiabilidade dos resultados do método proposto.

Considera-se que a cada época de decisão o sistema pode optar por uma decisão $a=\{0, 1, 2, ... 10\}$, a qual indica o número de máquinas que devem permanecer ligadas. A Taxa de chegadas no sistema é de sete requisições por unidade de tempo ($\lambda=7$), os dez servidores são idênticos e possuem uma taxa de atendimento de uma requisição por unidade de tempo ($\mu=1$), o custo de manter cada máquina ligada é de 30 por unidade de tempo ($cm_xl=30$), cada requisição que permanece no sistema gera um custo $h=10$ por unidade de tempo e toda vez que se liga ou desliga uma máquina um custo $clm_x = cdm_x = 10$ é acarretado ao sistema. Nesse exemplo não será considerado o ganho por finalizar uma requisição ($gf=0$).

Nesse sistema não existe um tamanho limite para a fila (tamanho máximo da fila = ∞). Desta forma, para se poder tratar esse sistema como um PMDTC é necessário admitir que a fila possui um limite máximo de *N* requisições, tal que a probabilidade de existirem N ou mais requisições no sistema pode ser negligenciada. Para esse trabalho escolheu-se um limite $N=100$, o qual, para os valores especificados de λ e μ, a probabilidade de existirem N ou mais requisições no sistema foi calculado como sendo da ordem de 10^{-15}.

O custo médio em longo prazo por unidade de tempo é de 319,65 e 319,4, para o custo calculado pelo método apresentado nesse trabalho e o custo calculado por Tijms (1994), respectivamente. Comparando os dois resultados, observa-se um erro de apenas 0,0078% quando se considera *N=100* em detrimento de utilizar uma formulação matemática mais apurada como demonstrado por Tijms (1994).

Nº de requisições	s1	S2	Nº de requisições	s1	S2	Nº de requisições	s1	S2
i = 0	0	3	i = 5	4	6	i = 11 ou 12	8	10
i = 1	1	4	i = 6	5	7	i = 13 ou 14	9	10
i = 2	2	4	i = 7	5	8	i ≥ 15	10	10
i = 3	2	5	i = 8 ou 9	6	9	-	-	-
i = 4	3	6	i = 10	7	10	-	-	-

Tabela 2. Política ótima para o sistema com dez servidores e uma fila

Quando se compara a política *R** obtida nesse trabalho com a política *R*'* de Tijms (1994), observa-se que as duas políticas são idênticas. Essa política é apresentada na Tabela 2, na qual os valores s1 e s2 indicam os limites máximo e mínimo de quantas máquinas devem permanecer ligadas a cada decisão. Isto é, se em uma época de decisão *m* máquinas estão ligadas e o número de requisições no sistema é *i*, então, se $m<s1$, deve-se mudar para a configuração com *s1* máquinas ligadas, se $s1 \leq m \leq s2$, o sistema deve permanecer com as *m* máquinas ligadas e se $m>s2$, o número de máquinas ligadas deve ser reduzido para s2.

Custo médio por manter requisições no sistema	86,977
Custo médio por manter as máquinas ligadas	218,863
Custo médio por ligar as máquinas	6,905
Custo médio por desligar as máquinas	6,905
Custo médio Total	319,65

Tabela 3. Média dos custos do sistema

Pelo método apresentado nesse trabalho também foram obtidos os custos parciais descriminados na Tabela 1, entretanto esses custos não foram apresentados por Tijms (1994) e desta forma não são passiveis de comparação. A média de cada um desses custos individuais são apresentados na Tabela 3. As probabilidades limites de cada configuração quando a política ótima da Tabela 2 é utilizada serão omitidas por questões de espaço no artigo.

5. Comentários Finais

Este artigo apresentou uma estratégia que associa dois aspectos fundamentais em avaliação de desempenho de sistemas complexos: (i) a especificação formal e com um alto grau de abstração provida pelos Statecharts e (ii) a capacidade de apresentar cenários, atribuindo custos às transições de estados, utilizando-se Processo Markoviano de Decisão.

Ao utilizar a especificação Statecharts para modelar o sistema, esta estratégia apresenta as vantagens de possibilitar a representação clara de atividades e/ou componentes paralelos do sistema, provendo meios de comunicação entre eles, permite também o uso da hierarquia entre estados para se "esconder" aspectos não essenciais à avaliação de

desempenho realizada e também possibilita que usuários do sistema menos familiarizados com as técnicas de avaliação de desempenho possam ter um entendimento mais claro e objetivo do modelo implementado. E dessa forma, possam sugerir mudanças que tornem o modelo mais coerente com o sistema real.

Já as vantagens de se utilizar o Processo Markoviano de Decisão consistem no fato de que este permite que seja realizada uma análise em longo prazo do sistema a respeito dos custos acarretados ao sistema devido às decisões escolhidas, o que nos possibilita obter uma política ótima que norteia as tomadas de decisões ao longo do tempo e o próprio fundamento matemático dos PMD, o qual estabelece que o próximo estado do sistema depende apenas do estado atual, permite a representação do sistema através de técnicas de especificação de alto nível, neste caso particular, Statecharts.

Por outro lado esta estratégia apresenta a limitação de que o número de estados do PMD está diretamente relacionado ao número de estados do modelo em Statecharts, não sendo passível de se realizar tratamentos matemáticos para a redução de estados como foi realizado por Tijms (1994). Contudo, após a etapa de modelagem, o especialista em desempenho pode realizar certas modificações na representação do sistema sem alterar a sua funcionalidade, para desta forma obter uma modelagem mais "enxuta" do sistema. Por exemplo, se no sistema da Figura 3 não houver necessidade de representar claramente quando as máquinas estão nos estados OC (ligado e ocioso) ou TR (ligado e trabalhando), os componentes M1 e M2 poderiam ser modelados apenas com dois estados básicos LIG e DESL, como apresentado na figura 5.

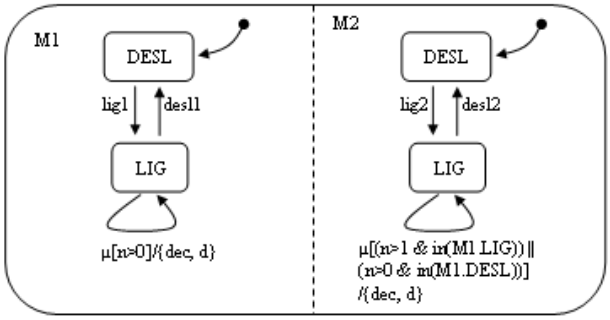

Figura 5. Modelagem das máquinas M1 e M2

O método apresentado neste artigo está sendo implementado na ferramenta PerformCharts, a qual foi desenvolvida por Vijaykumar (1999). Esta ferramenta foi desenvolvida em C++. Uma interface gráfica e mais amigável está em desenvolvimento. No entanto, para facilitar a especificação foi proposta uma linguagem PcML (PerfrmaCharts Markup Language) baseada em XML. Esta linguagem é interpretada por um script escrito em Perl e o resultado desta interpretação é um arquivo texto com código C++ que corresponde ao programa principal. Ao ligar este programa principal com o resto das classes, um executável é gerado a partir do qual é possível obter as medidas de desempenho para cenários desejados.

Referências

Beccuti, M., Franceschinis, G., e Haddad, S., (2007) *Markov Decision Petri Net and Markov Decision Well-Formed Net Formalisms*. In 28th Int. Conf. on application and theory of Petri nets and other models of concurrency (ICATPN) 2007, volume 4546 of LNCS, pages 43-62.

Chang, B. J., e Hwang, R. H., (2000) Hierarchical QoS Routing in ATM Networks Based on MDP Cost Function, *ICON '00: Proceedings of the 8th IEEE International Conference on Networks*, pp. 147–151.

Ghahnavie, A. R., e Firuzabad, M. F., (2006) Application of Markov Decision Process in Generating Units Maintenance Scheduling**,** *International Conference on Probabilistic Methods Applied to Power Systems*, PMAPS 2006. p 1-6.

Harel, D., (1987) Statecharts: A visual formalism for complex systems, *Sci. Computer Program*, vol. 8, no. 3, pp. 231-274.

Harel, D., Pnueli, A., Schmidt, J., e Sherman, R., (1987) On the Formal Semantics of Statecharts, *Proc. 2nd IEEE Symp. on Logic in Computer Science*, Ithaca, NY, p. 54-64.

Narasimha, C. Y., Kamath, M., e Sharda, R., (2007) A Semi Markov Decision Process Approach To E-mail Management in A Knowledge Work Environment, *IEEE International Conference on Automation Science and Engineering, 2007*. CASE 2007, p.1051 – 1056.

Pandana, C., e Liu, C. K. J. R., (2004) A near-Optimal Reinforcement Learning Scheme for Energy Efficient pointto-point Wireless Communications, *in IEEE Global Telecommunications Conference*, 2004 – GLOBECOM '04, pp. 763–767.

Puterman, L. M., (1994), *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Rodrigues, R C. M., (1998) *Inserção de distribuição do tipo fase em modelos markovianos de decisão*. Tese de doutorado, Instituto Nacional de Pesquisas Espaciais, INPE, Brasil,.

Sarmiento, A., e Riaño, G., (2005) *JMDP: An Object Oriented Framework for Modeling MDPs.* Informa Annual Meeting, San Francisco,USA.

Tijms, H. C., (1994) *Stochastic models: an algorithmic approach*, John Wiley & Sons.

Vijaykumar, N. L., (1999) *Statecharts: Their use in specifying and dealing with Performance Models*, Tese de Doutorado, Instituto Tecnológico de Aeronáutica, São José dos Campos, Brasil.