Execução Programada de Statecharts

João W. L. Cangussu [1]
e_mail: cangussu@brufms.bitnet
Departamento de Computação e Estatística - UFMS
C. P. 649 - Campo Grande , MS
Telefone (067) 787 3311 - Ramal 127

Paulo Cesar Masiero [2]
e_mail: masiero@icmsc.usp.br
José Carlos Maldonado [2]
e_mail: jcmaldon@icmsc.usp.br
Departamento de Ciências de Computação e Estatística - ICMSC - USP
C.P. 668 - São Carlos, SP
Telefone (0162) - 71 9144

Resumo

Este trabalho trata de uma técnica de validação de modelos, denominada "Execução Programada" e de sua integração com o ambiente Statechart Simulator (StatSim), que é um ambiente composto de ferramentas para edição e simulação de statecharts. Na "Execução Programada", o modelo comportamental do sistema sob desenvolvimento é simulado a partir de eventos gerados através de distribuições probabilísticas e é controlado por um programa que indica o que deve ser realizado em cada passo. Como resultado final, além da simulação dinâmica visual, dois relatórios são fornecidos ao usuário: um contendo análises estatísticas da execução e outro contendo o registro de todas as configurações atingidas, passo a passo.

Abstract

The two main concerns of this work are: a technique for model validation called "Programmed Execution" and its integration to the Statechart Simulator (StatSim) environment, wich is an integrated set of tools for edition and simulation of statecharts. In the "Programmed Execution", a system control model of a system under development is simulated from events generated according to probabilistics distributions and is controlled by a program that states what is to be done at each step. As results, besides the dynamic visual simulation, two reports are available: a statistical analysis about the simulation and a log file of all configurations reached during the simulation, step by step.

1-Introdução

Sistemas reativos podem ser definidos como sistemas que interagem direta ou indiretamente com o ambiente, recebendo e emitindo estímulos do mesmo, e que devem produzir os resultados corretos dentro de intervalos de tempo previamente especificados. Se o sistema atrasa em responder a um estímulo externo ou interno, o sistema falha [5]. Enquanto em um sistema transformacional o

1 - Apoio financeiro da Prefeitura
2 - Apoio financeiro do CNPq

interesse é voltado para o estado inicial e o estado final do sistema, em um sistema reativo todos os estados internos são importantes, devido à contínua interação com o ambiente [14].

Máquinas de Estados finitos, Redes de Petri e Statecharts são técnicas gráficas utilizadas na modelagem de sistemas reativos. Statecharts tem se mostrado bastante eficaz na especificação deste tipo de sistemas, motivo pelo qual vem sendo desenvolvidos alguns ambientes de apoio ao desenvolvimento de sistemas reativos baseados nesta técnica, entre os quais citam-se: Statemate, Argonaute e StatSim ([12], [15],[16]).

A maioria dos ambientes para especificação de sistemas reativos possui ferramentas para a execução interativa de modelos, seja de uma forma gráfica ou textual. Apesar de ser de grande valia para a verificação e validação de um modelo, a execução interativa (usuário interage diretamente com o sistema, acionando eventos e modificando variáveis e condições) explora o sistema de uma forma livre, ou seja, o usuário possui total controle sobre os acontecimentos no sistema. Muitas vezes é necessário, entretanto, que se possa observar o comportamento de um sistema reagindo sob condições aleatórias. Uma das maneiras de se obter essa visão é através da geração aleatória de eventos, condições e variáveis, controladas por uma linguagem, designada aqui por Linguagem de Controle de Execução (LCE). Outras formas de execução de modelos são a simulação não-interativa (usuário especifica um arquivo de entrada com os eventos a serem gerados em cada passo) e a simulação exaustiva, onde o sistema gera seqüências de disparos de eventos e alterações de variáveis e condições para atingir todas as configurações possíveis.

Execução programada é a simulação de um modelo (no caso statecharts) conduzida através de um programa escrito em uma linguagem de controle (no caso a LCE), com capacidade para gerar eventos de uma forma aleatória e alterar valores de variáveis e condições, fazendo com que a configuração do modelo possa ser alterada , ou não, durante a simulação.

A "Execução Programada" tem por objetivo complementar as técnicas de execução interativa, não-interativa e exaustiva, fornecendo resultados que são difíceis de serem extraídos destas técnicas [13]. Outra técnica de grande auxílio na verificação e validação de modelos é a prototipação, a qual dá ao usuário uma visão mais concreta do sistema final do que as técnicas de execução de modelos.

Apesar do ambiente StatSim ja possuir ferramentas que auxiliam a verificação e validação de modelos, tais como os algoritmos para detecção de deadlock, não determinismo, e outros, estes são baseados em uma árvore de alcançabilidade [3], e de acordo com o tamanho do modelo, essa árvore pode ultrapassar a capacidade de memória disponível, não sendo possível, nesses casos, utilizar os algoritmos citados. Esse foi um dos pontos que motivou o desenvolvimento da LCE, a qual não causa problemas de capacidade de memória e pode substituir esses algoritmos na busca das características de um modelo.

O objetivo deste trabalho é apresentar a Linguagem de Controle de Execução (LCE) e o seu interfaceamento com o ambiente StatSim, através do Módulo de Execução Programada (MEP). Este trabalho também apresenta o estudo de um caso com o objetivo de exemplificar a utilização da LCE.

Na segunda seção deste trabalho, apresenta-se uma introdução às principais características de Statecharts, seguida de uma breve descrição da evolução do ambiente StatSim; a terceira seção descreve a estrutura e a semântica da Linguagem de Controle de Execução (LCE); a quarta seção descreve as características e a interface do MEP (Módulo de Execução Programada); a quinta seção apresenta um modelo especificado em statechart, um programa para controle deste, escrito em LCE,

além do relatório estatístico gerado pela execução deste programa; por fim, a seção seis finaliza o trabalho, apresentando as conclusões deste.

2-Statecharts e o Ambiente StatSim

Os componentes básicos de um statechart são os estados, assim como as máquinas de estados finitos convencionais. Entretanto, estados podem ser embutidos em superestados criando uma hierarquia de estados. Os superestados podem ser de dois tipos: AND ou XOR. Os componentes de um estado AND são chamados componentes ortogonais e possuem a característica de que quando um sistema está em um estado AND, todos os seus componentes ortogonais também estão ativos. Ao contrário dos estados AND, quando um sistema está em um estado XOR, ele deve estar somente em um dos subestados. Na Figura 3 observa-se que "Mouse_Hand" é um estado AND com três componentes "Tops_number", "Clicks_number" e "Receive_signal", que são estados XOR.

O rótulo de uma transição pode ser definida por "e[c]/a", onde "e" é um evento, "c" é uma condição e "a" é uma ação, que pode ser um outro evento ou uma atribuição de variável. Se a ação for um evento, então este será ativado e poderá ser sentido nas componentes ortogonais, no mesmo passo (ou instante) em que foi gerado. Na Figura 3, suponha que o estado "Four" esteja ativo, desta forma se o evento "top" for disparado, então a transição de rótulo "top/rst" será executada, o que faz com que o evento "rst" seja sentido na componente "Clicks_number", que transiciona do estado corrente para "None". Portanto, através da geração de eventos (ações), os componentes ortogonais de um statechart podem se comunicar sincronamente.

A execução de um statechart está baseada em uma seqüência de passos de tempo onde, em cada passo, eventos gerados pelo ambiente são adicionados aos eventos gerados internamente como conseqüência da execução das ações, causando o disparo de transições. Na definição semântica percebe-se que o disparo de uma transição é instantâneo, enquanto que as atividades associadas a um estado leva certo tempo para serem executadas.

Statecharts possuem sintaxe e semântica formalmente especificadas [11]. Entre as várias características definidas, pode-se citar não determinismo e a capacidade de armazenar a última configuração de um estado (history). Existem vários eventos especiais, tais como: cr (current), ny (not yet), ch (changed), que juntamente com os eventos internos ex (exit), en (enterd), tr (true), fs (false) enriquecem os statecharts, possibilitando a representação de sistemas reativos dos mais variados tipos.

Diagramas de estados podem ser classificados como Máquinas Moore (nas quais as ações ou atividades estão associadas ao estado em que a máquina se encontra) ou Máquinas Mealy (ações ou atividades estão associadas às transições). Statecharts, entretanto, não podem ser enquadrados exclusivamente como uma Máquina Moore ou uma Máquina Mealy, mas como uma junção destas, pois se as ações podem ser geradas quando transições são executadas, ao se ativar um estado, isto significa que uma atividade poderá ser executada enquanto este estado estiver ativo. Portanto Statecharts unem os conceitos de Máquinas Moore e Mealy. Entretanto, devido à características síncronas dos Statecharts, a execução das ações é considerada instantânea, enquanto as atividades não o são, isto é, as atividades possuem um tempo associado à sua execução.

Statecharts são, portanto, uma técnica visual que possibilita especificar o aspecto comportamental de sistemas de tempo real de alta complexidade de uma forma clara e sucinta. Possui sintaxe e semântica formalmente definidas, o que facilita a verificação e validação de modelos.

O desenvolvimento de um ambiente, denominado StatSim, teve início com o projeto de uma Linguagem de Especificação de Statecharts, denominada LES, e um compilador para a mesma. Em seguida foram desenvolvidos um simulador textual, baseado na LES, e um editor gráfico de statecharts.

Com a implementação do compilador e do simulador para a LES, o usuário pode especificar o statechart de uma forma textual e os erros na especificação são detectados pelo compilador. Se o statechart for corretamente especificado, uma base de dados é gerada pelo compilador da LES, contendo todas as informações necessárias para que a simulação seja executada [9].

Com a base de dados gerada pelo compilador da LES, o usuário pode realizar a simulação textual do statechart, acionando eventos externos que farão com que o modelo seja reconfigurado. A semântica da simulação é definida por Harel em [11].

A ferramenta que seguiu ao desenvolvimento do analisador da LES e da simulação textual foi o editor gráfico, com o qual o usuário ao invés de especificar textualmente o statechart, pode editá-lo graficamente na tela. A base de dados criada pelo editor gráfico é basicamente a mesma criada pelo compilador da LES, a menos das informações gráficas. O editor gráfico faz uso também de partes do analisador da LES, como por exemplo os módulos que realizam a análise de expressões lógicas e aritméticas [1] [16].

Depois da implementação do editor gráfico, foi feita a integração deste com o simulador, ou seja, o editor gráfico foi interfaceado com o simulador e o resultado da simulação passou a ser animado e não textual, apesar de ainda ser possível analizar a simulação textualmente através do "log" gerado [17].

A partir do momento que a simulação passou a ser animada, o interesse no ambiente ficou voltado para a inserção de uma série de características dos statecharts que ainda não haviam sido implementadas. Desta forma foi implementada a reação em cadeia (broadcast), e todos os operadores e eventos internos de statecharts que não se encontravam no ambiente StatSim [10].

Um conjunto de algoritmos para a análise dos statecharts editados também foram desenvolvidos; esses algoritmos são baseados em uma árvore de alcançabilidade similar a árvore de alcançabilidade para Redes de Petri. Os algoritmos detectam deadlock, alcançabilidade entre configurações, uso de transições e validade de seqüência de eventos. A árvore de alcançabilidade gerada permite a análise de todas as possíveis configurações do modelo, ou seja, a análise exaustiva. Deve-se ressaltar entretanto, que as restrições de memória e de tempo impedem a análise exaustiva de modelos maiores e mais complexos [3].

Com isso, as características sintáticas e semânticas formalmente definidas por Harel [11] foram todas implementadas e o ambiente StatSim vem sendo usado para modelar sistemas, como por exemplo células de manufatura, aquecedores de ambiente e protocolos de comunicação de redes [7], entre outros, o que demonstra sua utilidade e versatilidade na modelagem do aspecto de controle de sistemas reativos diversos.

3-Linguagem de Controle de Execução (LCE)

Estrutura da LCE

A Linguagem de Controle de Execução (LCE) foi dividida, principalmente por motivos estruturais em três partes. A primeira parte diz respeito aos parâmetros globais, tais como o número de passos que serão simulados, a configuração inicial do modelo, o tipo de simulação (passo a passo ou contínua) e outros parâmetros. Esta parte relativa aos parâmetros pode ser omitida, e caso isto aconteça, os valores destes parâmetros são obtidos da interface do ambiente, se os mesmos foram especificados na interface, e caso contrário, os parâmetros recebem valores "defaults" previamente determinados.

A segunda parte da LCE trata da declaração de variáveis de controle e das declarações para geração de eventos. As variáveis de controle são variáveis inteiras utilizadas pelo usuário no controle dos programas escritos em LCE, portanto, tais variáveis não devem ser confundidas com as variáveis especificadas no statechart. As declarações relativas a eventos não significam que os mesmos estão sendo especificados, ou seja, o que estas declarações fazem é determinar a forma como os eventos serão gerados nos passos de simulação, mas não especificam novos eventos. Todos os eventos citados na declaração devem corresponder a eventos exitentes no statechart a ser executado. As formas como os eventos podem ser gerados são: distribuições probabilísticas (normal, exponencial e uniforme); periodicidade de eventos; probabilidade fixa de ocorrência; e arquivos de entrada [8].

A terceira parte da LCE executa o controle da simulação, iniciando com a palavra reservada "SIMULATION" e terminando com "END_SIMULATION". Esta parte da linguagem especifica as atividades que serão executadas nos passos, sendo composta de comandos para controle de passos e de outros comandos e funções que auxiliam o usuário nas tarefas de verificação e validação dos modelos especificados em Statecharts. A Tabela 1 apresenta uma breve descrição dos comandos, parâmetros e funções disponíveis na LCE, uma descrição mais detalhada com a sintaxe destes pode ser encontrada em [8].

Semântica da LCE

A semântica da LCE é baseada em uma semântica operacional similar à de Esterel [2] [4].

A semântica da LCE segue o formato abaixo, o qual indica que uma sentença, ou termina sua execução, ou deixa resíduos para serem executados. A sentença também pode alterar os conteúdos das memórias. Seja:

$$\langle \text{sentença,MES,MAP,MVC} \rangle \xrightarrow[i]{b} \langle \text{sentença',MES',MAP',MVC'} \rangle$$

Onde:

Tabela 1 - Comandos, Parâmetros e Funções da LCE

Comando	Descrição
Parâmetros Globais	
NUMBER_OF_STEPS	Determina o número de passos
SIMULATION_RESULT	Deatermina o resultado da simulação
SIMULATION_TYPE	Determina o tipo da simulação
STATECHART	Determina o modelo a ser simulado
INITIAL_CONFIGURATION	Determina a configuração inicial do modelo
Declaração de Variáveis de Controle e Eventos	
CONTROL_VAR	Declara variáveis de controle
READ_FILE	Coloca os eventos do arquivo na fila de eventos
EVENTS e1,...,en distribuição	Especifica uma dist. prob. para a geração de eventos
EVENTS e1,...,en X%	Especifica uma probabilidade de ocorrência dos eventos
EVENTS e1,...,en FROM	Especifica os passos de ocorrência dos eventos
Comandos e Funções	
AT STEP s1,...,sn	Especifica os comandos a serem exec. nos passos s1,...,sn
ALL STEPS	Especifica os comandos a serem exec. em todos os passos
FOR STEP	Especifica os comandos a serem executados nos passos detererminados no FOR
IF	Comando condicional
FOR	Comando de repetição
WHILE	Comando de repetição
SHOW_VAR	Exibe as variáveis do statechart
SHOW_CONTROL_VAR	Exibe as variáveis de controle
SHOW_EVENTS	Exibe os eventos dispardos no passo corrente
INTERACTIVE	Retrona para a execução interativa
EVENTS(parâmetro)	Dispara os eventos passados como parâmetro
AVAIL	Habilita os eventos passados como parâmetro
UNAVAIL	Desabilita os eventos passados como parâmetro
MESSAGE	Emite a mensagem especificada
SET_CONFIGURATION	Reconfigura o statechart
ACTIVE	Verifica se os estados especificados estão ativos
CONFIGURATION	Verifica se a configuração especificada está ativa

- sentença e sentença' são comandos e ou funções da linguagem e seus respectivos resíduos;
- MES e MES' correspondem à Memória de Especificação do Statechart (antes e depois de sentença ser executada) e contém informações sobre os estados, eventos e variáveis do statechart.
- MAP e MAP' correspondem à Memória de Atividades por Passo (antes e depois de sentença ser executada), contendo informações sobre os eventos a serem ativados em cada passo e as variáveis a serem exibidas, entre outras;
- MVC e MVC' correspondem à Memória das Variáveis de Controle (antes e depois de sentença ser executada) , as quais podem ser externas, se definidas pelo usuário, ou internas, se correspondem a variáveis para controle do número de passos, do passo atual, etc;
- b é um valor booleano que assume "true" se a sentença termina e "false" caso contrário; e
- i é um valor inteiro que indica o passo corrente de execução do modelo.

A descrição do conteúdo das memórias MAP, MES e MVC pode ser encontrada em [8], com detalhes de manipulação e especificações das mesmas.

A semântica de todos os comandos e funções da LCE foi especificada utilizando a notação dada acima. Aqui não será descrita a semântica dos comandos e funções da LCE, mas a mesma pode ser vista em [6].

4-Módulo de Execução Programada (MEP)

O Módulo de Execução Programada (MEP) consiste de elementos para edição de programas, interpretação dos mesmos e execução dos modelos de acordo com os programas escritos. O usuário escreve seus programas através de um editor de textos que está disponível no ambiente. Esses programas alimentam o interpretador da Linguagem de Controle de Execução (LCE), o qual possui acesso à base de dados dos statecharts para fazer certas verificações, tal como: verificar se um certo evento ou variável especificada no programa faz parte do statechart a ser executado.

Durante a interpretação das especificações (declarações) de eventos, o interpretador da LCE armazena os eventos a serem disparados em cada passo. Na simulação, a cada passo, o interpretador retira os eventos por ele armazenados e coloca-os na entrada do simulador. Assim, de acordo com os eventos , o modelo atinge uma nova configuração e o resultado da simulação será animado, textual (relatório de passos da simulação) ou ambos, dependendo da especificação do usuário.

O relatório de passos da simulação é alimentado a cada passo pelo interpretador e pelas mensagens armazenadas por este. O relatório estatístico é gerado no final da simulação , porém deve-se notar que grande parte dos cálculos vão sendo executados durante a

simulação. O relatório estatístico é gerado cada vez que um programa escrito na LCE for executado, independente de especificações do usuário.

O Módulo de Execução Programada foi desenvolvido em estações de trabalho SUN, utilizando C como linguagem de programação e as rotinas da biblioteca XView para desenvolvimento da interface. Em relação ao volume de programação, foram desenvolvidas 95 rotinas, as quais totalizam em torno de 5.000 (cinco mil) linhas de código fonte. Algumas rotinas foram aproveitadas do ambiente StatSim, com alterações.

Interface do MEP

A interface do MEP apresenta em primeiro lugar, um campo textual no qual o usuário deve especificar qual o programa fonte será executado. Os cinco campos a seguir correspondem aos parâmetros da LCE, ou seja, nesses campos especifica-se: o número de passos da simulação; o tipo de simulação; o resultado da simulação; a configuração inicial do statechart; e por fim, um campo que indica qual statechart será simulado.

Depois dos parâmetros, a interface apresenta um botão, o qual é utilizado para dar início à simulação. Ao lado do botão de início da simulação encontra-se, respectivamente, o botão que, se pressionado, executa a chamada de um editor para os programas fonte em LCE, e o botão que retorna à tela anterior do ambiente StatSim. A tela inicial do MEP é exibida na Figura 1.

Os próximos botões que aparecem são "Show Steps Report" e "Show Statistical Report" que acionados apresentam as telas para visualização do relatório de passos e do relatório estatístico. Se durante a simulação, o interpretador encontrar um dos comandos "SHOW_VAR", "SHOW_EVENTS" ou "SHOW_CONTROL_VAR", então a simulação é interrompida no final do passo corrente, e o menu de opções do MEP volta a ser visualizado na tela, nas seguintes condições: os únicos botões que podem ser acionados são "Continue Simulation" e os botões que se referem aos comandos "SHOW_CONTROL_VAR", "SHOW_VAR" e "SHOW_EVENTS".

Quando um comando "INTERACTIVE" é encontrado em um programa fonte, em um dos passos, então o simulador tem de interromper a execução para voltar à execução interativa. Nesse caso a simulação para no final do passo e a interface se apresenta ao usuário somente com os botões "Return", "Start New Simulation" e "Continue Simulation" possíveis de serem acionados. Se o usuário pressionar o botão "Return", então volta-se à simulação interativa e o usuário poderá fazer as alterações desejadas e depois retornar à simulação programada. Caso o usuário pressione o botão "Start New Simulation", então a interface retornará à tela inicial (Figura 1) e as estatísticas da simulação serão zeradas. Caso o usuário pressione o botão "Continue Simulation", então a simulação continuará a partir do passo no qual a mesma tinha sido interrompida. Na Figura 2 vê-se um exemplo da interface do MEP na situação descrita acima.

Um fato que deve ser notado é que durante a simulação a interface do MEP desaparece, para que dessa forma o usuário possa perceber a evolução animada do statechart na tela. Quando a simulação termina, a interface do MEP volta a aparecer na tela.

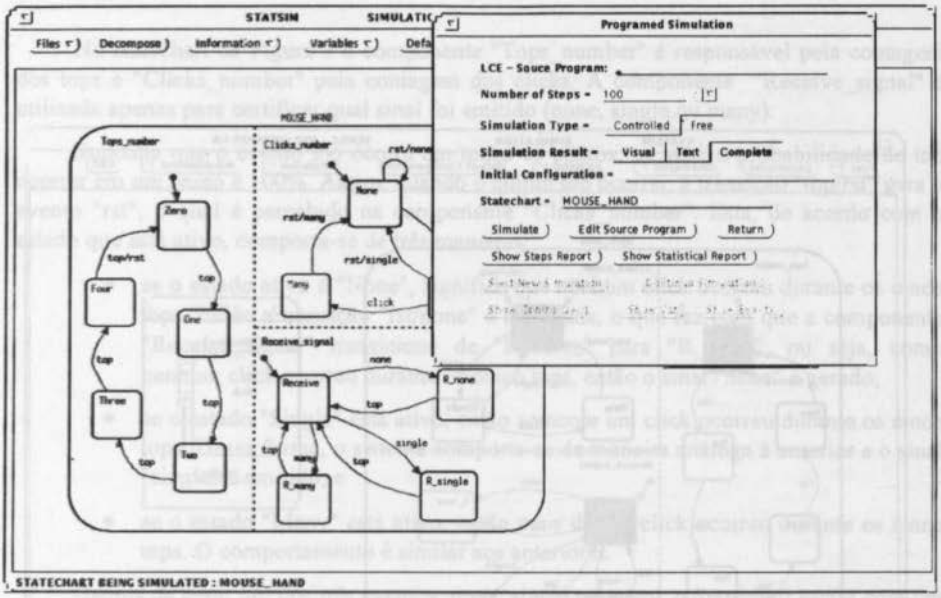

Figura 1 - Tela Inicial do MEP

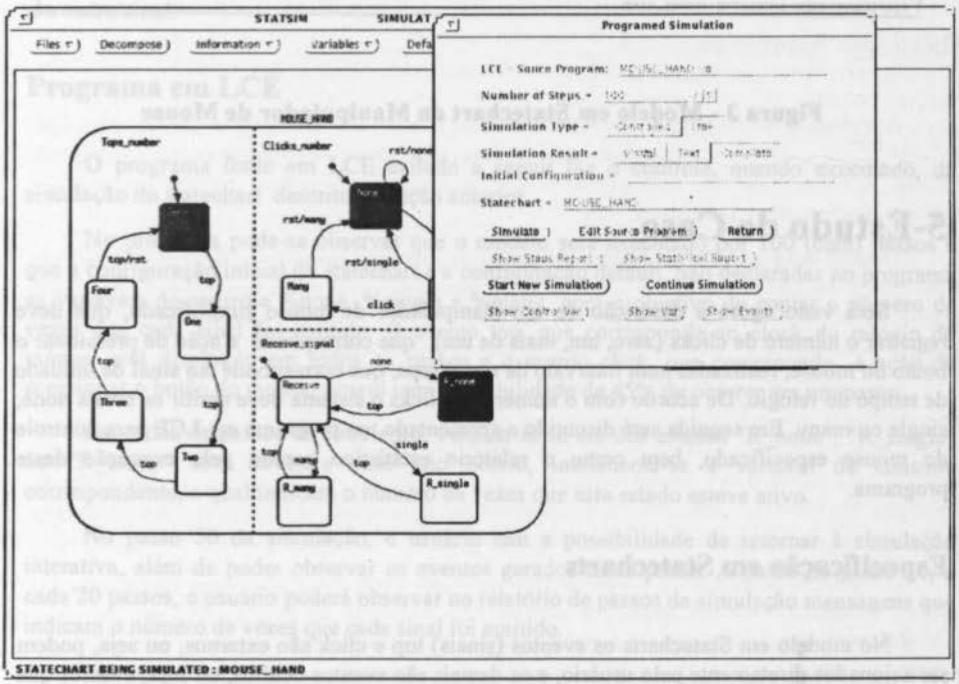

Figura 2 - Tela de Retorno à Simulação Interativa

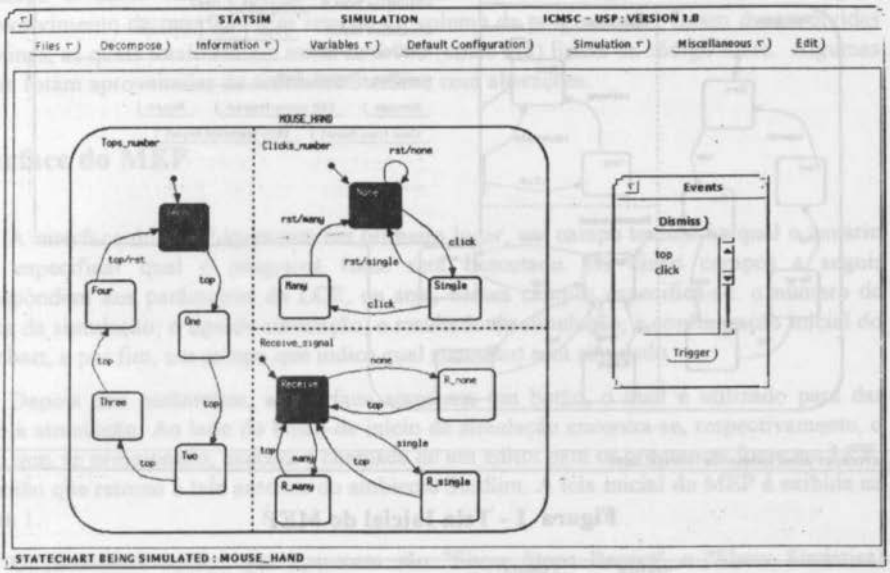

Figura 3 - Modelo em Statechart do Manipulador de Mouse

5-Estudo de Caso

Será visto agora a descrição de um manipulador de mouse simplificado, que deve registrar o número de clicks (zero, um, mais de um), que corresponde a ação de pressionar o botão do mouse, realizadas num intervalo de cinco tops, que corresponde ao sinal de unidade de tempo do relógio. De acordo com o número de clicks o sistema deve emitir os sinais none, single ou many. Em seguida será discutido e apresentado um programa em LCE para controle do mouse especificado, bem como o relatório estatístico gerado pela execução deste programa.

Especificação em Statecharts

No modelo em Statecharts os eventos (sinais) top e click são externos, ou seja, podem ser acionados diretamente pelo usuário, e os demais são eventos ocultos, ou seja, eventos que só podem ser gerados como a ação de uma transição.

No statechart da Figura 3 a componente "Tops_number" é responsável pela contagem dos tops e "Clicks_number" pela contagem dos clicks. A componente "Receive_signal" é utilizada apenas para certificar qual sinal foi emitido (none, single ou many).

Suponha que o evento top ocorre em todos os passos, ou seja, a probabilidade de top ocorrer em um passo é 100%. Assim, quando o quinto top ocorrer, a transição "top/rst" gera o evento "rst", o qual é percebido na componente "Clicks_number". Esta, de acordo com o estado que está ativo, comporta-se de três maneiras:

- se o estado ativo é "None", significa que nenhum click ocorreu durante os cinco tops. Então a transição "rst/none" é disparada, o que faz com que a componente "Receive_signal" transicione de "Receive" para "R_none", ou seja, como nenhum click ocorreu durante os cinco tops, então o sinal "none" é gerado;
- se o estado "Single" está ativo, então somente um click ocorreu durante os cinco tops. Dessa forma, o sistema comporta-se de maneira análoga à anterior e o sinal "single" é emitido; e
- se o estado "Many" está ativo, então mais de um click ocorreu durante os cinco tops. O comportamento é similar aos anteriores.

Depois de gerar um dos três eventos: none, single ou many, o Statechart estará com um dos estados "R_none", "R_single" ou "R_many" ativo. Porém, após o primeiro top, estes transicionarão para o estado "Receive", para que dessa forma possam receber a emissão de um outro sinal.

Programa em LCE

O programa fonte em LCE exibido a seguir faz o controle, quando executado, da simulação do statechart descrito na seção anterior.

No programa pode-se observar que o modelo será executado por 100 (cem) passos e que a configuração inicial do statechart é a configuração default. São declaradas no programa as variáveis de controle %none, %single e %many, com o objetivo de contar o número de vezes que cada sinal foi emitido. O evento top, que corresponde ao clock do relógio do mouse, será disparado em todos os passos e o evento click, que corresponde à ação de pressionar o botão do mouse, possui uma probabilidade de 45% de ocorrer em um passo.

Em todos os passos da simulação, verifica-se se um dos estados "R_none", "R_single" ou "R_many" está ativo, e caso isto ocorra, incrementa-se a variável de controle correspondente, a qual indicará o número de vezes que este estado esteve ativo.

No passo 50 da simulação, o usuário tem a possibilidade de retornar à simulação interativa, além de poder observar os eventos gerados neste passo. A partir do passo 20, a cada 20 passos, o usuário poderá observar no relatório de passos da simulação mensagens que indicam o número de vezes que cada sinal foi emitido.

```
GLOBAL PARAMETERS
   NUMBER_OF_STEPS = 100
   INITIAL_CONFIGURATION = DEFAULT
DECLARATIONS
   CONTROL_VAR %none , %single , %many
   EVENTS top FROM STEP 1 EACH 1 STEPS
   EVENTS click 45%
SIMULATION
   ALL STEPS
   BEGIN
   IF ACTIVE(R_none) THEN %none := %none + 1
      ELSE IF ACTIVE(R_single) THEN %single := %single + 1
         ELSE IF ACTIVE(R_many) THEN %many := %many + 1
   END
   AT STEP 50
   BEGIN
      INTERACTIVE
      SHOW_EVENTS
   END
   FOR STEP 20 EACH 20 STEPS DO
   BEGIN
      MESSAGE("Numero de sinais none emitidos = ",%none)
      MESSAGE("Numero de sinais single emitidos = ",%single)
      MESSAGE("Numero de sinais many emitidos = ",%many)
   END
END_SIMULATION
```

Relatório Estatístico

A execução do programa fonte descrito acima, além da simulação animada, gera dois relatórios: um contendo a configuração do modelo e informações sobre os eventos, variáveis, estados e mensagens em cada passo; e um segundo relatório que contém informações estatísticas do modelo executado.

Os primeiros resultados a serem exibidos no relatório estatístico, depois do nome do statechart e do número total de passos, são os resultados referentes aos estados. Para cada estado tem-se: o percentual de passos em que o estado esteve ativo; a situação final do estado (ligado ou desligado); e o número médio de passos consecutivos em cada estado.

Os eventos são analizados através do número de vezes que foram gerados e do número de passos em que estiveram disponíveis. A análise das transições é feita pela exibição do número de vezes que a transição foi disparada e do número de vezes que a transição foi avaliada. A comparação destes números dá o percentual de disparo por avaliação da transição. Os últimos resultados a serem apresentados no relatório estatístico são os valores médio e final das variáveis.

O relatório estatístico é utilizado para dar uma visão geral de certas estatísticas do sistema modelado. Mesmo quando o sistema não necessita de uma análise estatística, o relatório se constitui em um resumo da simulação, ou seja, nele pode-se ver se um evento foi gerado um número correto de vezes, se uma transição em que a condição sempre deveria ser verdadeira foi disparada 100% das vezes que foi avaliada, etc. Estes números podem ajudar a

STATECHART: MOUSE_HAND
NUMBER OF STEPS : 100

STATE	PERCENTAGE OF STEPS IN WICH WAS ACTIVE	FINAL SITUATION	AVERAGE NUMBER OF CONSECUTIVE STEPS IN THIS STATE
Zero	20.00	on	0.0000
One	20.00	off	0.0000
Two	20.00	off	0.0000
Three	20.00	off	0.0000
Four	20.00	off	0.0000
None	30.00	on	0.7222
Single	35.00	off	1.0588
Many	35.00	off	1.3333
Receive	83.00	of	3.6111
R_none	01.00	off	0.0000
R_single	02.00	off	0.0000
R_many	14.00	on	0.0000

EVENTS	NUMBER OF STEPS IN WICH THIS EVENT HAPPENED	PERCENTAGE OF STEPS IN WICH THIS EVENT WAS AVAILABLE
none	1	100.00
top	100	100.00
single	2	100.00
many	15	100.00
click	48	100.00
rst	20	100.00

TRANSITIONS	NUMBER OF STEPS IN WHICH THIS TRANSITION WAS FIRED	NUMBER OF STEPS IN WHICH THIS TRANSITION WAS EVALUATED	RATE OF FIRING SUCESS
top/rst	20	20	100.00
top	20	20	100.00
top	20	20	100.00
top	20	20	100.00
top	20	20	100.00
rst/none	1	1	100.00
rst/many	15	15	100.00
rst/single	2	4	50.00
click	15	15	100.00
click	17	18	94.44
top	14	14	100.00
many	15	15	100.00
top	2	2	100.00
single	2	2	100.00
top	1	1	100.00
none	1	1	100.00

VARIABLE	AVERAGE VALUE	FINAL VALUE
@none	0.267	1
@single	0.853	2
@many	10.518	14

Figura 4 - Relatório Estatístico Gerado pela LCE

detectar certos erros na especificação do statechart. Um exemplo de erro encontrado com a utilização da LCE é o statechart do manipulador de mouse (Figura 3), o qual possui o erro descrito a seguir.

Suponha que o statechart em questão possua a configuração {Four,None,Receive} e que os eventos "top" e "click" tenham sido gerado. Então na componente "Tops_number" a transição de rótulo "top/rst" será disparada. Na componente "Clicks_number" duas transições serão possíveis: "rst/none" e "click". Como as duas transições são inconsistentes entre si, somente uma poderá ser disparada (a escolha é aleatória). Se a transição de rótulo "click" for disparada, então terão ocorridos cinco tops e nenhum dos sinais "none","single" ou "many" terá sido emitido, o que se constitui em um erro na especificação. Este erro foi encontrado analisando-se o relatório estatístico gerado pela LCE, onde se percebe que a transição de rótulo "rst/single" foi avaliada 4(quatro) vezes e só foi disparada 2(duas) vezes. O relatório estatístico gerado é exibido na Figura 4.

6-Conclusão

Segundo Harel, ferramentas de desenvolvimento de software que não possuam capacidade de execução de modelos e de geração de código deverão desaparecer [13]. Seguindo essa filosofia, uma Linguagem de Controle de Execução (LCE) foi desenvolvida, a qual controla modelos especificados através de statecharts dentro do ambiente StatSim, complementando as técnicas de execução interativa, não interativa e exaustiva. O fato da LCE complementar a execução interativa e não interativa pode ser facilmente deduzido analisando-se as características da LCE [8]. Quanto ao fato da LCE complementar a execução exaustiva, isto se deve ao fato de que em muitos casos ela não pode ser aplicada, devido à restrições de tempo e de memória, podendo-se então executar o modelo através de um programa escrito em LCE e analisar os resultados obtidos.

Além da característica principal de controlar a execução de um modelo, a LCE possui outro aspecto importante, que são as características estatísticas da linguagem. Desta forma, a geração de eventos pode ser feita através de distribuições probabilísticas gerando-se, no final, o relatório estatístico.

Como pode ser visto em [7], a LCE controla a execução dos modelos, e o número de passos que um estado permanece ativo corresponde ao tempo que uma atividade demora para ser executada. Se um gerador de código estiver disponível, então a LCE poderá continuar a controlar a simulação, porém os eventos serão realmente gerados pelas atividades, as quais serão executadas de acordo com o código. O código gerado, entretanto, não representará o sistema final, mas provavelmente, estará bem mais próximo deste do que apenas a execução animada dos statecharts.

O desenvolvimento do MEP (Módulo de Execução Programada) aumentou o poder de validação e verificação dos modelos especificados através de statecharts dentro do ambiente StatSim.

Referências Bibliográficas

[1] BATISTA, J. E. S. Um Editor Gráfico para Statecharts. São Carlos , ICMSC-USP, 1991. (Dissertação de Mestrado)

[2] BERRY, G. ; GANTHIER, G. The Esterel Synchronous Programming Language: Design, Semantics, Implementation. França, INRIA, 1988. (Raports Technique, 842)

[3] BOAVENTURA, I. A. G. Propriedades Dinâmicas de Statecharts. São Carlos, ICMSC-USP, 1992. (Dissertação de Mestrado)

[4] BOUSSINOT, F. ; SIMONE, R. D. The Esterel Language. Proceedings of the IEEE, v. 79, n. 9, p.1293-303, 1991.

[5] BURNS, A. Scheduling Hard Real Time System: a review. Software Engineering Journal, v. 14, n. 3, p. 116-28, 1991.

[6] CANGUSSU, J. W. L. ; MASIERO, P. C. Sintaxe e Semântica da Linguagem de Controle de Execução. São Carlos, ICMSC-USP, 1992. (Relatório Técnico)

[7] CANGUSSU, J. W. L. ; MASIERO, P. C. Uma Linguagem para Execução Programada de Statecharts. In: Seminário Integrado de Software e Hardware, 19, Rio de Janeiro, 1992. Anais, Rio de Janeiro, SBC, p. 229-42, 1992.

[8] CANGUSSU, J. W. L. Execução Programada de Statecharts. São Carlos, ICMSC-USP, 1993. (Dissertação de Mestrado)

[9] FORTE, R. P. M. Uma Ferramenta de Apoio à Utilização de Statecharts para Especificação do Comportamento de Sistemas de Tempo Real Complexos. São Carlos, ICMSC-USP, 1991. (Dissertação de Mestrado)

[10] FURUUTI, R. ; MASIERO, P. C. Extensões ao Ambiente StatSim para permitir a Simulação de Micro-Passos. São Carlos, ICMSC-USP, Março 1992. (Relatório CNPq)

[11] HAREL, D. Statecharts: On the formal Semantics of Statecharts. In: Proceedings of the 2nd IEEE Symposium on Logic in Computer Science, Ithaca, N. Y., p. 54-64, 1987.

[12] HAREL, D. ; et al. STATEMATE: A Working Environment for the Development of Complex Reactive Systems. In: Proceedings of the Tenth International Conference on Software Engineering, Singapore, 1988, Washington, IEEE, 1988.

[13] HAREL, D. Biting the Silver Bullet - Toward a Brigther Future for System Development. IEEE Computer, v. 25, n. 1, p. 8-20, 1992.

[14] HUIZING, C. ; ROEVER, W. C. Introduction design choices in the semantics of Statecharts. Information Processing Letters, v. 37, p. 205-13, 1991.

[15] MARANINCHI, F. Argonaute: Graphical Description, Semantics and Verification of Reactive Systems by Using a Process Algebra. In: SIFAKIS, J. Automatic Verification Methods for Finite State Systems. Berlin, Springer, p. 38-53, 1989. (Lectures Notes in Computer Science, 407)

[16] MASIERO, P. C. ; FORTES, R. P. M. ; BATISTA, J. E. S. Edição e Simulação do Aspecto Comportamental de Sistemas de Tempo Real. In: Seminário Integrado de Software e Hardware, 18, Santos 1991. Anais, Santos, SBC, p. 45-61, 1991.

[17] TUTUMI, R. ; MASIERO, P. C. Simulação Visual de Statecharts. In: X Congresso de Iniciação Científica e Tecnológica em Engenharia, São Carlos 1991. Anais, São Carlos, CICTE, p. 669, 1991.