FCCE: Uma Família de Critérios de Teste para Validação de Sistemas Especificados em Estelle

Simone do Rocio Senger de Souza
Departamento de Informática
Universidade Estadual de Ponta Grossa-UEPG
srocio@uepg.br

José Carlos Maldonado
Deptº. de Ciências de Computação e Estatística
Universidade de São Paulo - ICMC/USP
jcmaldon@icmc.sc.usp.br

Sandra C.P.F. Fabbri
Departamento de Computação
Universidade Federal de São Carlos - UFSCar
sfabbri@dc.ufscar.br

Resumo

Estelle é uma técnica de descrição formal padronizada pela ISO e tem sido utilizada para a especificação de sistemas distribuídos e protocolos de comunicação. Este artigo propõe uma família de critérios de teste para a validação de sistemas especificados em Estelle, denominada Família de Critérios de Cobertura para Estelle – FCCE, estabelecendo-se mecanismos para quantificar a atividade de teste. A família FCCE pode ser utilizada tanto para a geração como para a avaliação de seqüências de teste. Além disto, esses critérios podem complementar as abordagens normalmente empregadas para a validação de especificações Estelle, como por exemplo, simulação. A representação de árvore de alcançabilidade para Estelle é proposta e utilizada para a aplicação dos critérios de teste. Esses aspectos são ilustrados utilizando-se a descrição em Estelle do protocolo Bit-Alternante.

Abstract

Estelle is a formal description technique standardized by ISO and have been used for specification of distributed systems and communication protocols. This paper proposes a family of coverage testing criteria for validation of systems specified in Estelle named Estelle Coverage Criteria Family (ECCF) establishing mechanisms to quantify the testing activity. The ECCF family can be used either to generate or to evaluate test sequences. These criteria aims at complementing the approaches used for validation of the Estelle specifications, for instance, simulation. Reachability tree representation for Estelle is proposed and used for application of the criteria. These aspects are illustrated using the Alternating-Bit Protocol Estelle specification.

1. Introdução

Teste de Software é uma atividade de Garantia de Qualidade que tem como objetivo a identificação de erros ainda não descobertos. Um aspecto importante da atividade de teste é o estabelecimento de uma estratégia que possa ser aplicada durante o desenvolvimento e manutenção do produto.

O sucesso dessa atividade está relacionado com a qualidade do conjunto de casos de teste. Nessa perspectiva, duas questões importantes são: "*Como selecionar casos de teste?*" e "*Como garantir que o produto foi suficientemente testado?*". Considerando que o teste exaustivo – em que o software é exercitado com todos os valores possíveis do domínio de entrada – é impraticável, critérios de teste são utilizados, os quais permitem selecionar um subconjunto do domínio de entrada preservando a probabilidade de revelar os erros existentes no produto. Esses critérios sistematizam a atividade de teste e podem também constituir uma

medida de cobertura dessa atividade [6, 10, 26].

Para aplicações críticas, erros podem trazer conseqüências desastrosas. Exemplos de tais aplicações são: sistemas de controle de tráfego, de aeronaves, de telecomunicações, controle bancário, monitoramento de pacientes, dentre outros, as quais possuem a característica de envolver diretamente seres humanos. Para essas aplicações, a qualidade é mais relevante pois falhas podem provocar perdas humanas e econômicas, tornando as atividades de especificação e teste bastante críticas. Esses softwares necessitam de metodologias mais rigorosas para o seu desenvolvimento, de forma a garantir que o software seja desenvolvido corretamente e que funcione de acordo com o esperado. As técnicas de especificação formal contribuem nesse sentido permitindo o desenvolvimento de especificações consistentes, completas e não ambíguas [9]. Máquinas de Estados Finitos (MEFs) [3], Statecharts [15], Redes de Petri [23], Estelle [18] e Lotos [7] são técnicas formais utilizadas para a descrição do aspecto comportamental de sistemas críticos.

Estelle é uma Técnicas de Descrição Formal (TDF) que descreve o sistema através de uma hierarquia de Máquinas de Estados Finitos Estendidas (MEFEs) que se comunicam através de canais bidirecionais. Estelle foi desenvolvida pela ISO (*International Organization for Standardization*) para especificação de sistemas distribuídos, serviços e protocolos de comunicação [18].

As TDFs fornecem facilidades para as atividades de verificação e validação da especificação. Sabe-se que, quanto mais cedo os erros forem detectados no processo de desenvolvimento do software, menos onerosa é a atividade de removê-los. Esse aspecto motiva a definição de critérios, métodos e estratégias para o teste de especificação auxiliando na identificação de enganos, defeitos e erros nas fases iniciais do processo de desenvolvimento.

São encontrados na literatura alguns trabalhos que apresentam métodos e critérios para a validação de especificações formais. São propostas algumas técnicas para seleção de seqüências de teste para especificações baseadas em MEFs e MEFEs, utilizadas principalmente no teste de conformidade de protocolos de comunicação [1, 6, 8, 32]. Os critérios de teste propostos por Ural [31], Ural e Yang [33], Probert e Guo [25], Fabbri et al. [12, 13, 14] e Souza et al. [27, 28] procuram utilizar o conhecimento adquirido no teste de programas, mapeando critérios de teste empregados no nível de programa para o nível de especificação. Essa abordagem é promissora e pode complementar as técnicas de simulação e análise de alcançabilidade normalmente empregadas para a validação de especificações baseadas em MEF, Redes de Petri, Statecharts, Estelle, dentre outras.

Na mesma linha desses trabalhos, este artigo apresenta a *Família de Critérios de Cobertura para Estelle – FCCE*, formada por critérios de teste que exploram os aspectos de Fluxo de Controle [5] da especificação. Esses critérios fornecem mecanismos para quantificar a atividade de teste, ou seja, é possível analisar a cobertura da especificação pelas seqüências de teste e também guiar a geração de seqüências de teste, adequadas por construção, a esses critérios. Desse modo, esses critérios podem complementar as atividades normalmente empregadas para validação de especificações baseadas em Estelle, como por exemplo, a atividade de simulação. Além disso, os conjuntos de casos de teste adequados a esses critérios podem ser utilizados também durante a realização de teste de conformidade.

Assim sendo, a FCCE contribui para a validação dos aspectos de comunicação, dos aspectos dinâmicos e do sincronismo do sistema especificado. A validação desses aspectos é viabilizada a partir da representação da especificação por meio da árvore de alcançabilidade. Essa representação é proposta com base na semântica de Estelle e considera algumas técnicas

de redução descritas em Barnard [4], de modo a minimizar o problema de explosão de estados.

Este artigo está organizado da seguinte forma: na Seção 2 são apresentados os conceitos básicos da técnica Estelle, os quais são fundamentais para o entendimento dos critérios de teste apresentados. Na Seção 3 a FCCE é definida. Na Seção 4 é descrita a árvore de alcançabilidade para Estelle e descrito como que os critérios FCCE podem ser aplicados para geração e avaliação de seqüências de teste. Na Seção 5 são apresentados os resultados da aplicação da FCCE na especificação do protocolo Bit-Alternante. Finalmente, na Seção 6 são apresentadas as conclusões e desdobramentos deste trabalho.

2. Estelle: Conceitos Básicos

Um sistema especificado em Estelle é estruturado em uma hierarquia de módulos que se comunicam através de troca de mensagens. O comportamento de cada módulo é descrito através de uma Máquina de Estados Finitos Estendida (MEFE) e utiliza, com algumas restrições e extensões, a linguagem Pascal para a descrição do sistema. As mensagens recebidas pelos módulos são armazenadas em filas (FIFO) de tamanho infinito e são processadas de acordo com as condições, prioridades e atrasos associados às transições da MEFE.

Em uma especificação Estelle podem ser definidas várias instâncias de módulos e conexões entre módulos. Essas instâncias podem ser criadas de forma estática – estabelecida quando a especificação é iniciada, e dinâmica – criada em tempo de execução, ou seja, as instâncias são criadas e/ou finalizadas com o disparo de transições. Além disso, existe o conceito de módulo pai e módulo filho, sendo que o módulo pai sempre tem prioridade sobre os módulos filhos [18]. Isso significa que se o módulo pai tiver transições aptas a ocorrerem em um passo, ele irá disparar uma de suas transições. Isso previne problemas de sincronização entre os módulos, visto que o módulo pai pode compartilhar as variáveis exportadas pelo módulo filho. Essa característica permite, por exemplo, que o módulo pai crie e finalize, dinamicamente, instâncias de módulos filhos.

Em Estelle, é possível descrever paralelismo síncrono, assíncrono e execução seqüencial dos módulos do sistema. Essas diferentes possibilidades de comunicação e sincronização são definidas através do atributo de cada módulo: *systemprocess*, *systemactivity*, *process* e *activity*. Módulos com atributo *systemprocess* ou *systemactivity* são chamados *system* e entre eles ocorre paralelismo assíncrono. Dentro de um módulo *systemprocess*, pode ocorrer paralelismo síncrono ou execução seqüencial, dependendo do atributo de seus módulos filhos, que pode ser *process* ou *activity*. Os filhos de um módulo *process* podem ser *process* ou *activity* e são executados sincronamente em paralelo; isso significa que uma transição de cada módulo é selecionada em cada passo e essas transições são executadas em paralelo. Por outro lado, os filhos de um módulo *activity* podem somente possuir o atributo *activity* e são executados seqüencialmente, ou seja, uma transição de um dos módulos é selecionada aleatoriamente para execução em cada passo. No caso de módulos *systemactivity*, seus módulos filhos podem possuir somente o atributo *activity* e com isso só ocorre execução seqüencial [9].

A semântica de Estelle permite que seja determinado o comportamento global do sistema especificado, definindo como as transições são selecionadas a partir dos módulos e quais são as regras para disparo dessas transições. O comportamento global de um sistema especificado em Estelle é definido pelo conjunto de todas as possíveis seqüências de *situações globais*, geradas a partir de uma situação inicial, sendo que o disparo de uma transição faz com que o

sistema mude de uma situação global para outra. Portanto, a semântica operacional de Estelle descreve de que forma essas seqüências de situações globais são geradas, ou seja, a maneira que as transições podem se intercalar para modelar o paralelismo entre os módulos do sistema. Como apontado por Budkowski e Dembinski [9], essa semântica operacional de Estelle auxilia na definição de ferramentas de apoio para a especificação dessa técnica, como por exemplo, simuladores, depuradores, interpretadores, compiladores, dentre outras.

Na Figura 1 é apresentada a arquitetura de um sistema especificado em Estelle. Nesse exemplo, a especificação *Sistema* possui três módulos *Usuário*, *A_B* e *Meio de Comunicação*. Existem canais de comunicação interligando o módulo *Usuário* com o módulo *A_B* (ponto de interação U) e interligando o módulo *A_B* com o módulo *Meio de Comunicação* (ponto de interação M). Esse protocolo é utilizado na Seção 5 para ilustrar a aplicação da Família de Critérios de Cobertura para Estelle.

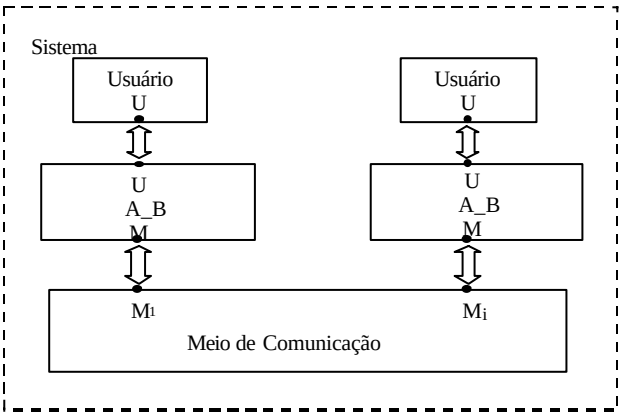


Figura 1. Arquitetura em Estelle do Protocolo *Bit-Alternante*.

3. FCCE: Família de Critérios de Cobertura para Estelle

A FCCE considera informações sobre o fluxo de controle da especificação e pode ser empregada para analisar a cobertura da especificação e também para guiar a geração de seqüências de teste, adequadas por construção, a esses critérios, ou seja, pode ser utilizada como critérios de adequação e como métodos de seleção de seqüências de teste.

A definição desses critérios de teste baseou-se em critérios definidos para programas paralelos e concorrentes: Taylor et al. [30], Yang e Chung [35], Chung et al. [2], Koppol e Tai [20] e Yang et al. [34]. Baseou-se também nos critérios de cobertura definidos para Statecharts [28]. Isso foi possível dado que a técnica Estelle permite modelar aspectos de paralelismo e concorrência da especificação, tais como, comunicação, sincronização e não determinismo.

A seguir são apresentados alguns conceitos que são essenciais para o entendimento dos critérios de cobertura definidos:

- ***Configuração:*** uma configuração C_i é um conjunto de estados do sistema que estão ativos em um passo da computação, sendo que C_0 é a configuração inicial. Em cada passo é suposto que os eventos associados à configuração atual sejam válidos e disparam as transições relacionadas, de modo que seja viável modelar o espaço de configurações possíveis do sistema.

- ***Caminho:*** é uma seqüência finita de configurações (C_0, C_i, C_j, …, C_m, C_k), $k \geq 1$, tal que a primeira configuração é a configuração inicial (C_0), e existe uma transição de C_i para C_j, $\forall\ C_i, C_j \mid 0 \leq i < k\ e\ j = i + 1$.
- ***Caminho Simples:*** é um caminho P tal que todas as configurações que compõem esse caminho, exceto possivelmente a primeira e a última configuração, são distintas.
- ***Caminho Livre de Laço:*** é um caminho simples P tal que todas as configurações são distintas, inclusive a primeira e a última.
- ***Caminho Reiniciável:*** é um caminho P em que a primeira e a última configuração do caminho correspondem à configuração inicial C_0, ou seja, são os caminhos que fazem com que o modelo retorne ao seu estado inicial.

Em se tratando de sistemas baseados em transições, a cobertura mínima desejável é executar todas as configurações de estados e todas as transições. Assim, dois critérios de teste são definidos:

1. Critério ***Todas-Configurações****:* requer que todas as configurações do modelo sejam percorridas no mínimo uma vez pelo conjunto de seqüências de teste.
2. Critério ***Todas-Transições****:* requer que todas as transições sejam executadas no mínimo uma vez pelo conjunto de seqüências de teste.

Chow [1] mostra que esses critérios não são apropriados para revelar erros típicos de especificações baseadas em MEF, tais como, *erros de transferência*, *erros de operação* e *erros de estados extras ou ausentes*. Isso pode ser considerado também para especificações baseadas em Estelle, visto que seus componentes básicos são MEFEs.

3. Critério ***Todos-Caminhos****:* requer que todos os caminhos sejam percorridos no mínimo uma vez pelo conjunto de seqüências de teste.

Do ponto de vista do teste estrutural, o critério *Todos-Caminhos* corresponde ao teste exaustivo, tornando-se impraticável, dada a possibilidade de existirem infinitos caminhos. Os critérios a seguir são mais rigorosos que os critérios *Todas-Transições* e *Todas-Configurações*, entretanto, menos dispendiosos que o critério *Todos-Caminhos*. Esses critérios estabelecem algum tipo de restrição para guiar a seleção de caminhos e também estabelecem uma "ponte" entre os critérios *Todos-Caminhos, Todas-Configurações* e *Todas-Transições:*

4. Critério ***Todos-Caminhos-k-C_0-Configuração****:* requer que todos os caminhos contendo k repetições da configuração C_0 sejam percorridos no mínimo uma vez pelo conjunto de seqüências de teste. Para $k = 2$ cada caminho reinicia a especificação uma vez e para $k > 2$, cada caminho reinicia a especificação $k-1$ vezes.

É importante observar que esse critério só é aplicado efetivamente para especificações que são reiniciáveis. Uma especificação é reiniciável quando, para cada configuração C_i alcançada a partir de C_0, existe uma seqüência de eventos que retorna a C_0.

5. Critério ***Todos-Caminhos-k-Configurações****:* requer que todos os caminhos contendo no máximo k repetições de cada configuração sejam percorridos no mínimo uma vez pelo conjunto de seqüências de teste.
6. Critério ***Todos-Caminhos-com-um-Laço****:* requer que todos os caminhos contendo no máximo 2 repetições de uma (somente uma) configuração C_i sejam percorridos no mínimo uma vez pelo conjunto de seqüências de teste.
7. Critério ***Todos-Caminhos-Simples****:* requer que todos os caminhos simples sejam percorridos no mínimo uma vez pelo conjunto de seqüências de teste.
8. Critério ***Todos-Caminhos-livre-Laço****:* requer que todos os caminhos livres de laços sejam percorridos no mínimo uma vez pelo conjunto de seqüências de teste.

As características específicas da técnica Estelle são consideradas por esses critérios de teste. Por exemplo, pode-se testar a comunicação entre os módulos através do recebimento de mensagens nos pontos de interação, informação que pode ser obtida pelas configurações da árvore. Ou seja, as seqüências de teste adequadas ao critério *Todas-Configurações* executam, no mínimo uma vez, todo recebimento possível de interações pelos pontos de interação.

Outra característica importante de Estelle que pode ser considerada pelos critérios de teste é o paralelismo entre os módulos da especificação. Conforme descrito anteriormente, Estelle permite três tipos de execuções dos módulos (componentes): execução seqüencial, paralela síncrona e paralela assíncrona. Considerando mais de um componente da especificação, as configurações representam as possíveis intercalações entre os estados de cada componente e, desse modo, o critério *Todas-Configurações* também executa todo paralelismo possível entre os módulos considerados. Dado que serão selecionados alguns componentes da especificação para análise e construção da árvore de alcançabilidade (Seção 4.1), a aplicação dos critérios de cobertura é restrita aos componentes selecionados, ou seja, a validação do paralelismo limita-se aos componentes selecionados.

Os componentes dinâmicos da especificação (módulos que podem ser criados e destruídos em tempo de execução pelos módulos hierarquicamente superiores) também podem ser validados utilizando os critérios de cobertura. Neste caso, cada configuração conteria os estados (estado local + conteúdo das filas) dos módulos selecionados e dos módulos criados dinamicamente. Para evitar a explosão de estados pode-se limitar o número de instâncias do mesmo módulo durante a construção da árvore de alcançabilidade. Da mesma forma que ocorre com a validação do paralelismo, a validação do aspecto dinâmico é restrito aos componentes selecionados para construção da árvore de alcançabilidade.

Os critérios FCCE estabelecem os requisitos de teste mínimos que precisam ser executados pelo conjunto de seqüências de teste T, de forma que T seja adequado a esses critérios de teste. Um conjunto de seqüências de teste T é adequado em relação a um critério C_R (descrito como C_R*-adequado)* se T satisfaz ou executa todos os requisitos de teste impostos por C_R [26].

Durante o estabelecimento dos requisitos de teste é possível derivar um conjunto de seqüências de teste adequado, por construção, aos critérios de teste. Um conjunto de seqüências de teste T é adequado por construção a um critério C_R quando os requisitos de teste estabelecidos por C_R guiam a geração de T, ou seja, T é construído de forma a cobrir cada requisito de C_R.

4. Caracterização dos Requisitos e Seqüências de Teste dos Critérios FCCE

Nesta seção são descritos como os requisitos de teste dos critérios FCCE podem ser facilmente obtidos a partir da representação da especificação por meio da árvore de alcançabilidade, definida a seguir. Os procedimentos para a geração da árvore de alcançabilidade para Estelle são apresentados em Souza [29].

A árvore de alcançabilidade permite representar o comportamento dinâmico da especificação (ou do sistema), descrevendo todos os estados que podem ser alcançados a partir de um estado inicial. A explosão de estados é um fator que inviabiliza a sua utilização e, desse modo, alguns trabalhos apresentam propostas de redução da árvore de alcançabilidade, durante a sua construção [4, 20, 21]. Conforme será visto a seguir, algumas dessas propostas são consideradas na definição da árvore de alcançabilidade para especificações em Estelle.

4.1. Árvore de Alcançabilidade para Estelle

Conforme mencionado, a árvore de alcançabilidade consiste em gerar o comportamento possível do sistema modelado. A partir da configuração inicial do sistema, todas as transições disparáveis são representadas, juntamente com as configurações alcançadas. Para cada nova configuração inserida na árvore, são obtidas as transições disparáveis e as configurações alcançadas e assim, sucessivamente, até que todas as configurações alcançáveis, a partir da configuração inicial, sejam representadas.

A árvore de alcançabilidade tem sido utilizada com sucesso em algumas áreas de aplicação, como por exemplo, em protocolos de comunicação para validação e análise de propriedades do modelo [24], sendo uma das principais técnicas para análise de Redes de Petri [22]. Tem sido utilizada também para análise de sistemas especificados através de MEF, MEFE, Statecharts e Estelle [4, 16, 17, 19, 21].

Em especificações Estelle cada configuração (*estado global)* é formada pelos *estados locais* dos módulos do sistema, estrutura hierárquica dos módulos, ligações entre os módulos, variáveis, conteúdo dos canais de comunicação, e pelas transições selecionadas para execução. O *estado local* de um módulo *P* possui a notação *(Sp, tp)*, sendo que *Sp* é um dos estados de *P* e *tp* é uma transição oferecida por *P*. A partir de um estado *Sp*, mais de uma transição pode estar apta a disparar mas apenas uma é escolhida aleatoriamente (não determinismo), ou seja, pode existir mais de um estado local para *Sp*.

Para minimizar a explosão de estados da árvore, as seguintes técnicas de redução são consideradas durante a sua construção: *a) nós duplicados* – representa configurações repetidas na árvore. Quando uma configuração já existente é inserida, ela é considerada apenas um *link* para a primeira ocorrência dessa configuração, não sendo geradas novamente suas configurações sucessoras; *b) stubborn sets* – trata transições disparáveis que são independentes entre si, ou seja, transições que podem ser disparadas em qualquer ordem antes de ser obtida a próxima configuração. Ao invés de serem consideradas todas as possíveis combinações dessas transições, apenas uma das possibilidades é considerada, chamada *stubborn set;* e *c) conjunto de componentes* – considera apenas alguns componentes do sistema modelado durante a construção da árvore de alcançabilidade. A obtenção da árvore de alcançabilidade selecionando apenas alguns componentes do sistema é muito útil em se tratando de especificações de protocolos de comunicação compostos de várias camadas. Em geral, as camadas do protocolo são especificadas (e implementadas) separadamente, descrevendo-se as interações entre elas.

Desse modo, cada configuração C_i representa um possível estado do(s) componente(s) selecionado(s), possuindo as seguintes informações: estado $E(C_i)$ da(s) MEFE(s) do(s) componente(s) e conteúdo das filas $C(C_i)$ dos pontos de interação do(s) componente(s). O conteúdo de $C(C_i)$ é expresso pelas primitivas de comunicação recebidas.

Para ilustrar a construção da árvore de alcançabilidade, a especificação do Protocolo *Bit-Alternante* é utilizada (descrita na Seção 5). O módulo que especifica a funcionalidade do protocolo Bit_Alternante (*A_B)* é o componente escolhido. Esse módulo possui dois pontos de interação: ponto de interação *U* – por onde é enviada a primitiva *receiveresponse* e por onde são recebidas as primitivas *sendrequest* e *receiverequest* do módulo *Usuário;* e ponto de interação *M* – por onde é enviada a primitiva *datarequest* e recebida a primitiva *dataresponse* do módulo *Rede.* A MEFE desse componente possui dois estados: *estab* e *ackwait*. Na Tabela 1 são apresentadas as primitivas de entrada e de saída para cada transição dos módulos da especificação. Para definição de cada configuração da árvore foi necessário considerar

também um *buffer B* contedo as mensagens recebidas pelo protocolo (módulo *A_B)*. Isso foi feito devido aos aspectos funcionais desse protocolo: as transições *recrespa* e *recrespe* só estarão aptas a disparar se a primitiva *receiverequest* estiver na fila *U* e se o *buffer B* não estiver vazio. *B* armazena a mensagem recebida e que será enviada para o usuário destino. Assim, para esse exemplo, cada configuração da árvore de alcançabilidade possui os seguintes elementos:

C_i = *[estado, U, N, B]* em que:
*estado = {*estab, *ackwait}*
U = {0, sendrequest, receiverequest}
M = {0, dataresponse}
B = {0,1}

Na Figura 2 é apresentada a árvore de alcançabilidade do componente *A_B* do protocolo *Bit-Alternante*. São consideradas as transições dos componentes não selecionados (dos componentes *Usuário* e *Rede*) que são responsáveis por enviar entradas para as transições do componente *A_B*. Essas transições são representadas por arcos tracejados para diferenciar das transições do componente *A_B* e têm a descrição da primitiva que enviam. Essas transições indicam o recebimento das primitivas de comunicação nas filas *U*, *N* e no *buffer B* Existem 15 configurações possíveis e são representadas também *configurações velhas*, as quais aparecem com o símbolo ↑, o que facilita a visualização da árvore.

O controle do tamanho do árvore de alcançabilidade permite que os critérios FCCE possam ser aplicados para especificações mais complexas. Entretanto, a construção da árvore considerando alguns componentes da especificação tem como desvantagem a impossibilidade de representar o comportamento global, pois o comportamento observável fica restrito aos componentes selecionados.

Tabela 1. Primitivas de Entradas e de Saídas para as Transições dos Módulos do Protocolo Bit-Alternante.

Transições das MEFEs		**Primitivas de Entrada**	**Primitivas de Saída**
Módulo Usuário	*Sendit*	-	*U.sendrequest*
	Reqit	-	*U.receiverequest*
	Recvit	*U.receiveresponse*	-
Módulo A_B	*Send*	*U.sendrequest*	*N.datarequest*
	Getdatae	*D.dataresponse*	*N.datarequest*
	Goodack	*D.dataresponse*	-
	Recrespa	*U.receiverequest*	*U.receiveresponse*
	Getdataa	*D.dataresponse*	*N.datarequest*
	Badack	*D.dataresponse*	-
	Recrespe	*U.receiverequest*	*U.receiveresponse*
	Tossack	*D.dataresponse*	-
	Retrans	-	*N.datarequest*
Módulo Rede	*req1*	*N[1].datarequest*	*N[2].dataresponse*
	req2	*N[2].datarequest*	*N[1].dataresponse*
	loss1	*N[1].datarequest*	-
	loss2	*N[2].datarequest*	-

C0 = [estab, 0,0,0]
C1 = [estab, sendrequest, 0,0]
C2 = [estab, receiverequest, 0,0]
C3 = [estab, receiverequest, 0,1]
C4 = [estab, sendrequest, dataresponse, 0]
C5 = [estab, receiverequest, dataresponse, 0]
C6 = [estab, sendrequest, 0,1]
C7 = [ackwait, receiverequest, 0,0]
C8 = [ackwait, 0, dataresponse, 0]
C9 = [ackwait, 0, dataresponse, 1]
C10 = [ackwait, receiverequest, dataresponse, 0]
C11 = [ackwait, 0, 0, 0]
C12 = [ackwait, receiverequest, dataresponse, 1]
C13 = [estab, 0, 0, 1]
C14 = [ackwait, 0, 0, 1]
C15 = [ackwait, receiverequest, 0, 1]

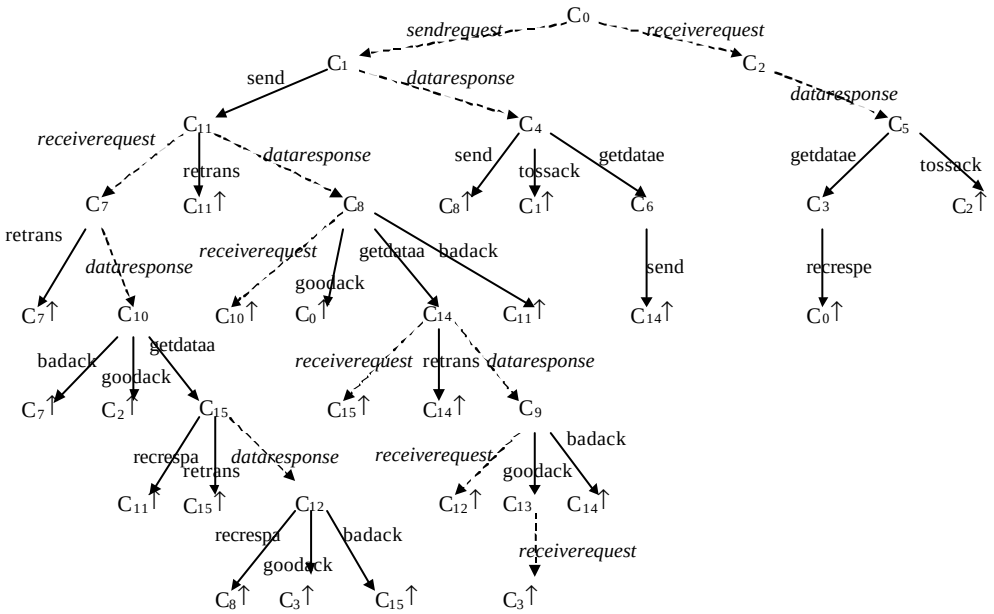


Figura 2. Árvore de Alcançabilidade para o Protocolo Bit-Alternante Especificado em Estelle.

4.2. Obtenção dos Requisitos de Teste dos Critérios FCCE

Os requisitos de teste são as informações mínimas que precisam ser percorridas pelo conjunto de casos de teste adequado a um critério de teste. Para obter cada requisito de teste r_i dos critérios FCCE, é definido um algoritmo para cada critério de teste que determina quais configurações da árvore irão fazer parte de r_i. Esses algoritmos são descritos em Souza [29]. De acordo com a definição dos algoritmos de cada critério FCCE, a árvore de alcançabilidade é percorrida em profundidade até que todos os requisitos dos critérios sejam obtidos. Durante a construção da árvore dois conjuntos são definidos: *SC:* conjunto de configurações da árvore e *T:* conjunto de transições da árvore. Esses conjuntos são utilizados pelos critérios de teste para obtenção dos requisitos de teste. Na Figura 3 é ilustrada uma síntese do algoritmo para obtenção dos requisitos de teste para o critério *Todos-Caminhos-k-C_0-Configuração.*

```
Cada requisito ri é obtido percorrendo-se a árvore a
partir de C0 e verificando cada nova configuração Cj:
    k = 2; {número de vezes que C0 pode ocorrer em ri}
    n = 0; {número de vezes que C0 ocorre em ri}
    enquanto (n < k) faça
       Cj = obtem_configuração();
       enquanto (Cj ≠ C0) faça
            ri = ri ∪ Cj;
       fim-enquanto;
       n = n + 1;
       se (n ≤ k) então
           ri = ri ∪ Cj;
       fim-enquanto;
    TR = TR ∪ {ri};
```

Figura 3. Algoritmo para Obtenção dos Requisitos de Teste para o Critério *Todos-Caminhos-k-C_0-Configuração*.

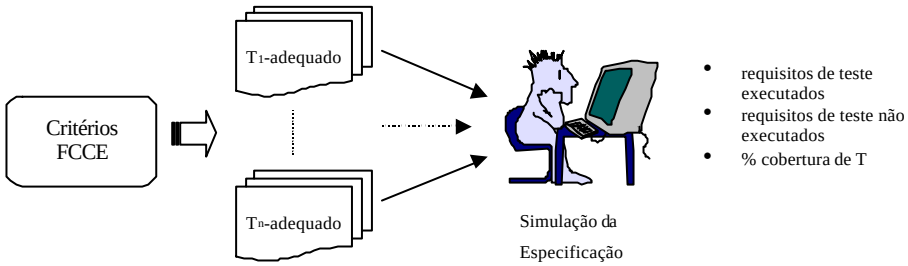
4.3. Geração de Seqüências de Teste Adequadas a FCCE

Os critérios FCCE podem ser empregados como critérios de adequação e como métodos de seleção de seqüências de teste. Esses aspectos são ilustrados na Figura 4. Na Figura 4a é ilustrada a utilização da FCCE para geração de seqüências de teste T_i adequadas. A partir das seqüências T_i's, uma possível aplicação é utilizá-las para alimentar a simulação interativa ou *batch* da especificação, funcionando como conjuntos iniciais de seqüências de teste para a simulação. O usuário pode então avaliar o comportamento do modelo com os conjuntos T_i e inserir novas seqüências de teste conforme o andamento da simulação. Outra aplicação seria utilizar as seqüências T_i's durante a realização de testes de conformidade.

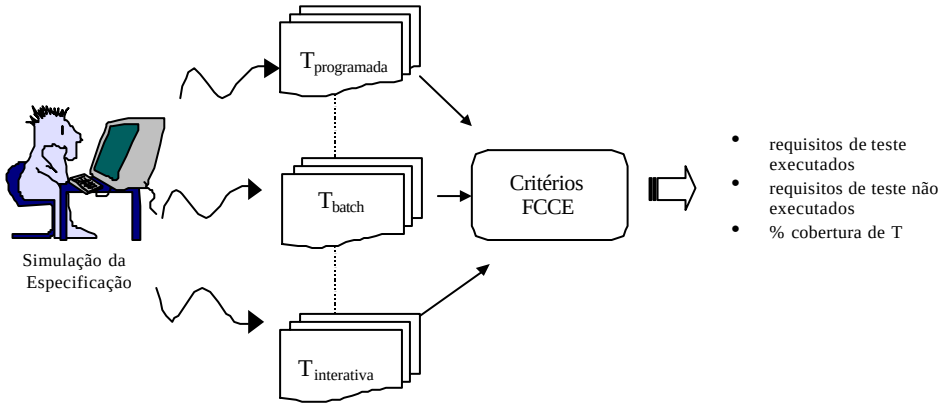
A utilização dos critérios FCCE como critérios de adequação é ilustrada na Figura 4b. Nesse caso, os critérios são utilizados para avaliar a cobertura de seqüências de teste geradas pela simulação. A partir de um conjunto T, obtido durante a simulação, o usuário pode aplicar os critérios FCCE e obter a porcentagem de cobertura para as configurações, transições, enfim, para qualquer um dos critérios definidos anteriormente. Esse aspecto pode ser explorado também para seqüências de teste adequadas a outros critérios de teste (Teste de Mutação, por exemplo), para seqüências de teste geradas aleatoriamente ou geradas manualmente. Nos dois casos (método de seleção e critério de adequação), as seguintes informações podem ser obtidas: requisitos de teste executados pelas seqüências de teste, requisitos que não foram executados pelo conjunto de seqüências de teste aplicado e porcentagem de cobertura obtida em relação aos critérios de cobertura.

As seqüências de teste adequadas por construção podem ser geradas à medida que a árvore é percorrida para obtenção dos requisitos de teste. Quando a configuração C_j é incluída em r_i a transição percorrida para encontrar C_j é incluída em um conjunto t_i que corresponde à seqüência de teste que percorre o requisito r_i. Analogamente à obtenção do conjunto de requisitos de teste, um conjunto de seqüências de teste é obtido, o qual é formado pelas seqüências de teste t_i que percorrem os requisitos r_i's.

O conjunto de requisitos de teste para os critérios *Todas-Transições* e *Todas-Configurações* é obtido sem percorrer a árvore de alcançabilidade. Através dos conjuntos T e SC, gerados durante a construção da árvore, é possível obter os requisitos desses critérios. Dessa forma, para obtenção de seqüências de teste adequadas por construção a esses critérios, foram definidos algoritmos para percorrer a árvore de modo a obter seqüências de teste que executem, no mínimo uma vez, cada requisito de teste desses critérios de teste.



a) Critérios de cobertura e a Geração de Seqüências de Teste.



b) Critérios de Cobertura e a Avaliação de Seqüências de Teste.

Figura 4. Os Critérios de Cobertura na Validação de Especificações.

5. Aplicação dos Critérios de Teste FCCE: Protocolo Bit_Alternante

A especificação do protocolo *Bit-Alternante* [18] é empregada para ilustrar a aplicação dos critérios por ser bastante conhecida e utilizada. A estrutura da especificação em Estelle desse protocolo é composta de um módulo principal, chamado de *Sistema*, o qual possui três módulos filhos (Figura 1):

- ***Usuário***: descreve os usuários conectados que podem enviar e receber mensagens.
- ***A_B****:* descreve o protocolo *Bit-Alternante*, o qual procura fornecer uma comunicação confiável sobre um meio de comunicação não confiável. Esse protocolo associa um bit (0 ou 1) em cada mensagem para determinar quando essas mensagens devem ser retransmitidas.
- ***Rede****:* descrição do meio de comunicação que recebe as mensagens do usuário e envia-as para o usuário destino.

Para aplicação dos critérios FCCE é considerada a árvore de alcançabilidade da Figura 2, ou seja, é considerado somente o módulo *A_B*. Na Tabela 2 são apresentados o total de requisitos e de seqüências de teste gerado para o módulo *A_B*. O critério *Todos-Caminhos* gera um número infinito de caminhos e por isso não foi considerado. O critério *Todos-Caminhos-k-Configurações* também não foi considerado porque gera um número muito elevado de requisitos de teste. Para a maioria dos critérios de teste, o número de seqüências de teste gerado é igual ao número de requisitos de teste porque cada requisito de teste corresponde a um caminho distinto na árvore de alcançabilidade. Observa-se também

que o número de requisitos de teste gerado para cada critérios de teste é bastante elevado, exigindo um número elevado de seqüências de teste para executar cada requisito de teste. Na Tabela 3 são apresentados alguns requisitos e seqüências de teste dos critérios FCCE aplicados.

Tabela 2. Total de Requisitos e de Seqüências de Teste Gerado para os Critérios FCCE para a Especificação Estelle do Protocolo *Bit-Alternante*.

Critérios FCCE	**Requisitos de Teste**	**Número de Seqüências deTeste**
Todos-Caminhos	infinito	–
Todos-Caminhos-k-C_0-Configurações	17652	17652
Todos-Caminhos-k-Configurações	não aplicado	–
Todos-Caminhos-com-um-Laço	292	292
Todos-Caminhos-Simples	46	46
Todos-Caminhos-livre-Laço	39	39
Todas-Transições	37	22
Todas-Configurações	16	04
TOTAL	18082	18055

Tabela 3. Subconjunto de Requisitos de Teste (tr) e Seqüências de Teste (ts) dos Critérios FCCE para a Especificação Estelle do Protocolo *Bit-Alternante.*

Critérios FCCE	**Requisitos de Teste e Seqüências de Teste**
Todos-Caminhos-k-C_0-Configurações	Tr = {(c0, c1, c11, c7, c7, c10, c2, c5, c3, c0), (c0, c1, c11, c7, c7, c10, c2, c5, c2, c5, c3, c0) …} Ts = {(sendit, send, reqit, retrans, req1, goodack, req1, getdatae, recrespe), (sendit, send, reqit, retrans, req1, goodack, req1, tossack, req1, getdatae, recrespe) …}
Todos-Caminhos-com-um-Laço	Tr = {(c0, c1, c11, c7, c7, c10, c2, c5, c3), (c0, c1, c11, c7, c7, c10, c15, c12, c8, c14, c9, c13, c3) …} Ts = {(sendit, send, reqit, retrans, req1, goodack, req1, getdatae), (sendit, send, reqit, retrans, req1, getdataa, req1, recrespa, getdataa, req1, goodack, reqit) …}
Todos-Caminhos-Simples	Tr = {(c0, c1, c11, c7, c10, c2, c5, c3, c0), (c0, c1, c11, c7, c10, c15, c12, c8, c0) …} Ts = {(sendit, send, reqit, req1, goodack, req1, getdatae, recrespe), (sendit, send, reqit, req1, getdataa, req1, recrespa, goodack)…}
Todos-Caminhos-livre-Laço	Tr = {(c0, c1, c11, c7, c10, c2, c5, c3), (c0, c1, c11, c7, c10, c15, c12, c8, c14, c9, c13, c3) …} Ts = {(sendit, send, reqit, req1, goodack, req1, getdatae), (sendit, send, reqit, req1, getdataa, req1, recrespa, getdataa, req1, goodack, reqit)…}
Todas-Transições	Tr = {(c0,c1), (c1,c11), (c0,c2), (c2,c5), (c11,c7), (c1,c4), (c4,c8), (c5,c3), (c7,c7), (c11,c11), (c11,c8), (c8,c10), …} Ts = {(sendit, send, reqit, retrans), (sendit, send, retrans), (sendit, send, req1, reqit), (sendit, req1, send), …}
Todas-Configurações	Tr = {c0, c1, c2, c11, c4, c5, c7, c8, c6, c3, c10, c14, c15, c9, c12, c13} Ts = {(sendit, send, reqit, req1, getdataa, req1), (sendit, send, req1, getdataa, req1, goodack), (sendit, req1, getdatae), (reqit, req1, getdatae)}

Os critérios FCCE foram aplicados também como mecanismos para avaliar a cobertura de seqüências de teste geradas pela simulação. Dessa forma, demonstra-se como esses critérios podem auxiliar a melhorar a atividade de validação de especificações em Estelle. A especificação Estelle do protocolo *Bit_Alternante* foi simulada utilizando-se a ferramenta

EDT e as seqüências de teste geradas pela simulação foram avaliadas utilizando-se os critérios FCCE.

A ferramenta *EDT* fornece um método automático para obtenção de implementações distribuídas, na linguagem C, a partir de especificações em Estelle [11]. Dentre as ferramentas que fazem parte da *EDT*, o simulador/depurador *Edb* permite que o usuário observe o comportamento dinâmico da especificação. O usuário pode controlar, observar e rastrear a execução da especificação através de comandos do simulador. A especificação pode ser simulada interativamente ou de maneira controlada. Na simulação interativa, o usuário conduz a simulação escolhendo as transições entre as transições aptas a disparar ou determinando um módulo a partir do qual as transições serão selecionadas. Na simulação controlada, o usuário pode definir, por meio de uma linguagem, um programa que controla a simulação. Para geração das seqüências de teste, foi utilizado um programa disponível na ferramenta *EDT*, que gera as seqüências aleatoriamente. Esse programa executa 5 vezes 20 transições escolhidas aleatoriamente, reiniciando o simulador para tentar obter seqüências de transições diferentes.

A partir das seqüências de teste geradas pelo simulador, analisou-se a porcentagem de requisitos dos critérios FCCE que são executados ou percorridos por essas seqüências. Os resultados são apresentados na Tabela 4. A porcentagem de cobertura é calculada da seguinte forma:

$$\frac{\text{Número de Requisitos executados por } \mathbf{T}}{\text{Total de Requisitos}} * 100$$

sendo que ***T*** representa o conjunto de seqüências de teste gerado pela simulação.

Devido ao rigor dos critérios de cobertura, os quais exigem que seqüências de teste percorram caminhos específicos na árvore de alcançabilidade a cobertura foi muito baixa. A cobertura poderia ser melhorada se o tamanho (número de transições) de cada seqüência fosse maior. Outra situação seria disparar manualmente as transições de modo a percorrer um número aceitável de requisitos de cada critério de teste.

Esses dados ilustram a utilização dos critérios FCCE para guiar a atividade de simulação de especificações Estelle. Nesse contexto, os critérios poderiam ser utilizados para mensurar a cobertura das seqüências de teste obtidas durante a simulação em relação a esses critérios de teste. Esse processo poderia ser realizado incrementalmente selecionando os critérios relevantes e tentando simular novas situações até que todos os requisitos dos critérios de teste escolhidos fossem satisfeitos, contribuindo assim para efetivamente aprimorar a atividade de teste de especificações em Estelle.

Tabela 4. Resultado da Avaliação da Cobertura das Seqüências de Teste Geradas pela Simulação em relação à FCCE para a Especificação do Protocolo *Bit-Alternante*.

Critérios FCCE	**# Requisitos Exercitados**	**Porcentagem de Cobertura**
Todos-Caminhos-k-C_0-Configurações	01	0
Todos-Caminhos-com-um-Laço	01	0.3
Todos-Caminhos-Simples	02	4.3
Todos-Caminhos-livre-Laço	0	0
Todas-Transições	13	35.1
Todas-Configurações	10	62.5
TOTAL	27	17.0

6.Conclusões

Este artigo apresentou uma família de critérios de teste, chamada de *Família de Critérios de Cobertura para Estelle – FCCE*, para a validação de especificações baseadas em Estelle. A FCCE é composta por critérios que focalizam os aspectos de fluxo de controle da especificação e pode ser empregada tanto para geração como para avaliação de seqüências de teste. De acordo com o conhecimento dos autores, este é o primeiro trabalho que procura utilizar, no contexto de Estelle, a árvore de alcançabilidade para auxiliar na geração de seqüências de teste.

Um problema inerente com a representação através de árvore de alcançabilidade é a explosão de estados, fato que compromete a aplicação dos critérios de teste. Neste sentido, a árvore de alcançabilidade para Estelle foi definida considerando algumas técnicas de redução durante a construção da árvore, a exemplo de outros trabalhos nesta linha, como por exemplo: Masiero et al. [21], Koppol e Tai [20] e Barnard [4].

A FCCE pode complementar outras abordagens de validação empregadas para Estelle, como simulação, por exemplo. Esse aspecto foi ilustrado utilizando os critérios para avaliar seqüências de teste geradas pela simulação.

Dois aspectos que precisam ser explorados são o custo e a eficácia da FCCE. O custo avalia o esforço necessário para aplicação dos critérios de teste e a eficácia avalia a capacidade dos critérios para revelar os erros existentes. Estes aspectos estão sendo investigados nos trabalhos atuais do grupo.

É importante observar que o conjunto de seqüências de teste adequado para testar a especificação pode ser empregado para a condução dos testes de conformidade da implementação. Nesse sentido, pretende-se investigar o relacionamento entre esses níveis de abstração: especificação e implementação.

Como trabalhos futuros nessa linha de pesquisa, pretende-se investigar os seguintes aspectos:

- Estender os critérios de cobertura definindo critérios de Fluxo de Dados, na mesma linha de pesquisa de Ural e Yang [33] e Yang et al. [34]. Isto requer que sejam adicionadas informações sobre o fluxo de dados da especificação na árvore de alcançabilidade;
- Implementação de uma ferramenta de apoio à aplicação desses critérios de teste;
- Condução de estudos empíricos para avaliar o custo e a eficácia em revelar erros da FCCE, de forma a produzir subsídios para determinação de uma estratégia incremental para aplicação desses critérios na prática;
- Realização de estudos comparando a FCCE com o critério Teste de Mutação, o qual também tem sido explorado no contexto de Estelle [27];
- Explorar a aplicação desses critérios no contexto de outras técnicas para descrição formal de protocolos de comunicação, tais como SDL e Lotos.

Agradecimentos

Os autores agradecem à CAPES, FAPESP, CNPq pelo apoio dado ao Grupo de Pesquisa em Engenharia de Software do ICMC/USP e à Universidade Estadual de Ponta Grossa (UEPG). Agradecem também ao Prof. Dr. Stanislaw Budkowski (Institut National des Telecommunications - France) que gentilmente cedeu uma versão da ferramenta EDT para a condução das pesquisas.

Referências

[1] Chow, T.S. Testing Software Design Modeled by Finite-State Machines. IEEE Transaction on Software Engineering, SE-4(3), maio, 1978.

[2] Chung, C-M.; Shih, T.K.; Wang, Y-H.; Lin, W-C.; Kou, Y-F. Task Decomposition Testing and Metrics for Concurrent Programs. In: Fifth International Symposium on Software Reliability Engineering (ISSRE'96), p.122-130, 1996.

[3] Gill, A. Introduction to the Theory of Finite-State Machine. New Jork, McGraw-Hill, 1962.

[4] Barnard, J. COMX: A Design Methodology Using Communicating X-Machines. Information and Software Technology, 40, p.271-280, 1998.

[5] Beizer, B. Software Testing Techniques. 2a Edição, Van Nostrand Reinhold, N. York, 1990.

[6] Bochmman, G.V.; Petrenko, A. Protocol Testing: Review of Methods and Relevance for Software Testing. In: International Symposium on Software Testing and Analysis (ISSTA'94), ACM-Software Engineering Notes, p.109-124, 1994.

[7] Bolognesi, T.; Brinksma, E. Introduction to the ISO Specification Language Lotos. Computer Networks and ISDN Systems, 14, p.25-59, 1987.

[8] Bourhfir, C.; Dssouli, R.; Aboulhamid, E.M. Automatic Test Generation for EFSM-Based Systems. Publication Departamentale #1043, 1996 (disponível em: www.umontreal.ca/labs/teleinfo/PubListIndex.html).

[9] Budkowski, S.; Dembinski, P. An Introduction to Estelle: A Specification Language for Distributed Systems. Computer Network and ISDN Systems, 14, p.3-23, 1987.

[10] DeMillo, R.A.; Lipton, R.J.; Sayward, F.G. Hints on Test Data Selection: Help for the Practicing Programmer. IEEE Computer, abril, 1978.

[11] EDT. Estelle Development Toolset (versão 4.1). Institut National des Télécommunications, Evry, França, 2000 (Disponível em: www.alix.int-evry.fr/~stan/edt.html).

[12] Fabbri, S.C.P.F.; Maldonado, J.C.; Delamaro, M.E.; Masiero, P.C. Mutation Analysis Testing for Finite State Machine. In: Fifth International Symposium on Software Reliability Engineering (ISSRE'94), Califórnia, p.220-229, novembro, 1994.

[13] Fabbri, S.C.P.F.; Maldonado, J.C.; Masiero, P.C.; Delamaro, M.E. Mutation Analysis Applied to Validate Specifications Based on Petri Nets. In: 8th IFIP Conference on Formal Descriptions Techniques for Distributed Systems and Communication Protocol (FORTE'95), p.329-337, Montreal, Canadá, 1995.

[14] Fabbri, S.C.P.F.; Maldonado, J.C.; Sugeta, T.; Masiero, P.C. Mutation Testing Applied to Validate Specifications Based on Statecharts. In: International Symposium on Software Reliability Engineering (ISSRE'99), 1999.

[15] Harel, D.; Pinnel, A.; Schmidt, J.P.; Sherman, R. On the Formal Semantics of Statecharts. In: 2nd IEEE Symposium on Logic in Computer Science, Thaca, New York, 1987.

[16] Huang, C-M.; Hsu, J-M. An Incremental Protocol Verification Method. The Computer Journal, 37(8), 1994.

[17] Huang, C-M.; Hsu, J-M.; Lai, H-Y.; Huang, D-T.; Pong, J-C. An Estelle-Based Incremental Protocol Design Systems. J. Systems Software, 36, p.115-135, 1997.

[18] ISO/TC97/SC21/WG1/DIS9074. Estelle – A formal Description Technique Based on an Extended State Transition Model. 1987.

[19] Jirachiefpattana, A.; Lai, R. EVEN: A Software Environment for Estelle Specification Verification. Journal of Systems Software, 39, p.119-143, 1997.

[20] Koppol, P.V.; Tai, K-C. An Incremental Approach to Strutural Testing of Concurrent Software. In: International Symposium on Software Testing and Analysis (ISSTA'96), ACM-Software Engineering Notes, p.14-23, 1996.

[21] Masiero, P.C.; Maldonado, J.C.; Boaventura, I.G. A Reachability Tree for Statecharts and Analysis of Some Properties. Information and Software Technology, 36(10), p.615-624, 1994.

[22] Murata, T. Modeling and Analysis of Concurrent Systems. Handbook of Software Engineering, Van Nostrand Reinhold Electrical, New York, 1984.

[23] Peterson, J.L. Petri Nets. Computing Surveys, vol. 9(03), setembro, 1977.

[24] Pezzè, M.; Taylor, R.N.; Young, M. Graph Models for Reachability Analysis of Concurrent Programs. ACM Trans. on Software Engineering and Methodology, 4(02), abril, 1995.

[25] Probert, R.L.; Guo, F. Mutation Testing of Protocols: Principles and Preliminary Experimental Results. Protocol Test Systems, III, Ed. by I. Davidson and D.W. LitwackNorth-Holland, p.57-76, 1991.

[26] Rapps, S.; Weyuker, E.J. Selecting Software Test Data Using Data Flow Information. IEEE Transaction on Software Engineering, vol.11(04), p.367-375, abril, 1985.

[27] Souza, S.R.S.; Maldonado, J.C.; Fabbri, S.C.P.F.; Lopes de Souza, W. Mutation Testing Applied to Estelle Specifications. Software Quality Journal, vol. 8(04), to appear, artigo selecionado do 33rd Hawaii International Conference on System Sciences, Mini-track: Distributed Systems Testing, Maui, Hawaii, 4-7 de janeiro, 2000a.

[28] Souza, S.R.S.; Maldonado, J.C.; Fabbri, S.C.P.F.; Masiero, PC. Statecharts Specifications: A Family of Coverage Testing Criteria. In: Conferência Latino Americana de Informática (CLEI2000), Cidade de México, México, 18-22 de setembro, 2000b.

[29] Souza, S.R.S. Validação de Especificações de Sistemas Reativos: Definição e Análise de Critérios de Teste. Tese (Doutorado) - Instituto de Física de São Carlos da Universidade de São Paulo (IFSC/USP), São Carlos, SP, dezembro, 2000.

[30] Taylor, R.N.; Levine, D.L.; Kelly, C.D. Structural Testing of Concurrent Programs. IEEE Transaction Software Engineering, 18(3), março, 1992.

[31] Ural, H. Test Sequence Selection Based on Static Data Flow Analysis. Computer Communications, v.10(5), p.234-242, outubro, 1987.

[32] Ural, H. Formal Methods for Test Sequence Generation. Computer Communications, 15(5), junho, 1992.

[33] Ural, H.; Yang, B. A Test Sequence Selection Method for Protocol Testing. IEEE Transaction on Communications, 39(4), abril, 1991.

[34] Yang, C-S.; Souter, A.L.; Pollock, L.L. All-Du-Path Coverage for Parallel Programs. In: International Symposium on Software Testing and Analysis (ISSTA'98), ACM-Software Engineering Notes, p.153-162, 1998.

[35] Yang, R-D.; Chung, C-G. Path Analysis Testing of Concurrent Programs. Information and Software Technology, 34(1), janeiro, 1992.