Modelagem e Verificação Formal de Sistemas de Informação Baseados em Componentes

Hyggo Almeida[1]*, Elthon Oliveira[2], Nádia Barbosa[2], Frederico Bublitz[2], Leandro da Silva[1], Angelo Perkusich[1]

[1]Departamento de Engenharia Elétrica – Universidade Federal de Campina Grande
Av. Aprigio Veloso, 882 - Bodocongo, 58109-970 - Campina Grande - PB - Brasil

[2]Departamento de Sistemas e Computação – Universidade Federal de Campina Grande

{hyggo,angelo,leandro}@dee.ufcg.edu.br, {easo,nmb,fmb}@dsc.ufcg.edu.br

Resumo. *Neste artigo propõe-se a utilização de um arcabouço denominado COMPOR-CPN para a modelagem e verificação formal de sistemas de informação baseados em componentes. O arcabouço é uma modelagem em Redes de Petri Coloridas da especificação de componentes COMPOR. Descreve-se a aplicação do arcabouço na modelagem e verificação de um sistema de informação baseado em componentes para a gerência de comunidades virtuais.*

Abstract. *This paper describes the application of a framework named COMPOR-CPN to the formal modelling and verification of component based information systems. The framework is a Colored Petri Net modelling of the COMPOR component model specification. We describe the usage of the COMPOR-CPN to model and verify an information system for managing virtual communities.*

1. Introdução

A construção de Sistemas de Informação com base em componentes reutilizáveis vem se tornando cada vez mais comum na medida em que requisitos como flexibilidade, facilidade de manutenção e evolução tornam-se indispensáveis à redução de custos e confiança no funcionamento e na autonomia de tais sistemas.

Contudo, sistemas baseados em componentes, principalmente aqueles construídos com base em componentes de terceiros, possuem sérias restrições quanto à confiança no seu funcionamento, o que pode afetar de forma significante um requisito primordial para os sistemas de informação atuais: robustez.

Para abordar este problema, técnicas de modelagem e verificação formal vêm sendo largamente utilizadas no contexto de Engenharia de Software e Métodos Formais. Ao contrário da utilização de testes, a verificação formal garante que a especificação do sistema baseado em componentes é correta para todos os cenários possíveis, assegurando a confiança no funcionamento do sistema.

Neste artigo propõe-se a utilização de um arcabouço denominado COMPOR-CPN para a modelagem e verificação formal de sistemas de informação baseados em componentes. O arcabouço é uma modelagem em Redes de Petri Coloridas [Jensen 1997] (CPN - *Coloured Petri Nets*) da especificação de componentes COMPOR [Almeida et al. 2004b]. CPN têm sido aplicadas em vários domínios, tendo suporte de um conjunto de ferramentas computacionais denominado Design/CPN

*Bolsista CNPq no Programa de Doutorado do Departamento de Engenharia Elétrica - COPELE/DEE

(`http://www.daimi.au.dk/designCPN/`) para edição dos modelos e diversos tipos de análise como, por exemplo, simulação e verificação de modelos.

Descreve-se a utilização do arcabouço na modelagem e verificação de um sistema de informação baseado em componentes para gerência de comunidades virtuais. As diretrizes para modelagem e verificação, assim como os modelos de CPN necessários para a instanciação do arcabouço neste contexto são apresentados.

O restante do artigo está organizado da seguinte forma: na Seção 2. é apresentada uma introdução informal à CPN; na Seção 3. apresenta-se o arcabouço COMPOR-CPN; na Seção 4. descreve-se a aplicação do arcabouço COMPOR-CPN ao sistema de informação para gerência de comunidades virtuais; alguns trabalhos relacionados são discutidos na Seção 5.; e na Seção 6., são apresentadas as considerações finais.

2. Redes de Petri Coloridas

Redes de Petri Coloridas são uma ferramenta matemática com uma representação gráfica de modelagem aplicável a vários tipos de sistemas como, por exemplo, sistemas concorrentes, assíncronos e distribuídos [Jensen 1997]. Usando CPN, um sistema deve ser modelado através da descrição de seus estados e dos eventos que levam à mudança destes estados. Tais estados são associados a *lugares* e *marcações*, e os eventos a *transições* [Jensen 1997]. Cada lugar pode ter em um dado momento um conjunto de fichas correspondente ao seu conjunto de cor (tipo de dado associado ao lugar), denominado marcação do lugar. A marcação da rede é representada pela marcação de todos os lugares da rede em um dado instante de tempo.

A notação gráfica de CPN é muito útil para a visualização do sistema de um modo mais amigável, facilitando eventuais correções de especificação antes da fase de implementação. Os seguintes elementos fazem parte da notação gráfica de CPN: os *lugares* são representados por elipses/círculos; *transições* são representadas por retângulos; fichas representam as *marcações*; setas representam *arcos*, os quais ligam um lugar a uma transição (ou vice-versa) e possuem uma inscrição referente ao *peso* relacionado a uma quantidade de fichas; e *conjunto de cores*, que determinam o tipo da ficha associado a um determinado lugar (inteiro, string etc).

Uma vez que as transições determinam a mudança de estados, é o disparo destas que guia o comportamento do modelo CPN. Para isso, existem algumas regras que devem ser obedecidas para que o disparo de uma transição ocorra: (i) uma transição t está habilitada a disparar se cada um dos lugares de entrada P_i possuir pelo menos $w(P_i, t)$ fichas, onde $w(P_i, t)$ representa o peso do arco indo de P_i a t; (ii) uma transição habilitada poderá disparar assim que os eventos relacionados a ela ocorrerem; (iii) o disparo de uma transição t remove $w(P_i, t)$ fichas de cada lugar de entrada P_i e adiciona $w(t, P_0)$ fichas a cada lugar de saída P_0 de t. Além disso, transições podem ter condições de disparo explicitamente definidas como guardas. Sendo assim, além das condições normais de disparo, uma transição só estará habilitada caso sua condição de guarda seja verdadeira.

Além dos conceitos básicos, o conceito de *lugar de fusão* é importante para o entendimento do restante do artigo. Lugares de fusão são lugares diferentes que compartilham a mesma marcação. Dois ou mais lugares de fusão formam um conjunto de fusão. Por exemplo, caso uma transição de entrada ou saída coloque ou retire n fichas de um lugar P_i, serão colocadas ou retiradas n fichas de todos os outros lugares pertencentes ao conjunto de fusão.

3. O arcabouço COMPOR-CPN

O arcabouço COMPOR-CPN é uma modelagem em CPN da especificação do modelo de componentes COMPOR (CMS) [Almeida et al. 2004b], a qual provê mecanismos para a redução do impacto da inserção, remoção ou alteração dos componentes de uma aplicação, inclusive em tempo de execução.

De acordo com a CMS, um sistema baseado em componentes é composto por duas entidades principais: contêineres e componentes funcionais. Os componentes funcionais implementam as funcionalidades do sistema, disponibilizando-as como serviços. Os componentes funcionais não são compostos por outros componentes, ou seja, não possuem "componentes-filhos". Os contêineres, por sua vez, não implementam funcionalidades, apenas gerenciam o acesso aos serviços dos seus "componentes-filhos" [Almeida et al. 2004b].

O arcabouço COMPOR-CPN é a modelagem da especificação CMS, permitindo que qualquer sistema baseado em componentes, de acordo com esta especificação, possa ser modelado e verificado formalmente de uma maneira simples e sistemática. COMPOR-CPN é uma extensão do modelo CPN para verificação da CMS, apresentado em [de Almeida et al. pear]. Em tal modelo, tem-se como propósito apenas uma verificação da arquitetura definida na CMS, com foco na interação entre os componentes considerando a hierarquia de contêineres.

Para a definição do COMPOR-CPN, não apenas a interação entre os componentes é importante, mas também o comportamento interno de cada componente e seu impacto sobre o restante da arquitetura. Na Figura 1, ilustra-se o módulo principal do COMPOR-CPN - a modelagem genérica do *componente*. Além deste módulo, existem os módulos referentes ao *contêiner* e ao *ambiente*, os quais podem ser encontrados em [de Almeida et al. pear].

Para extensão do modelo proposto em [de Almeida et al. pear], foram acrescentados dois lugares de fusão (*INPUT* e *OUTPUT*). Além disso, a transição *ProcLocal*, que antes apenas consumia uma ficha simulando a execução interna do componente, foi renomeada para *ToBeProcessed*. *ToBeProcessed* envia a requisição para ser processada pelas redes que modelam os componentes do sistema. As redes dos componentes, por sua vez, devem possuir dois lugares referentes à entrada (requisição) e saída (retorno). O lugar de entrada deve pertencer ao conjunto de fusão de *INPUT* (*C_In*) e o lugar de saída deve pertencer ao conjunto de fusão de *OUTPUT* (*C_Out*).

Por ser um lugar de fusão, ao ser inserida uma ficha em *INPUT* da página *Component*, a mesma ficha será inserida em todos os lugares de entrada de todos os componentes modelados. Sendo assim, deve haver um filtro para que cada componente modele apenas o comportamento dos serviços que implementa. Para isso, é necessário definir uma transição de entrada no componente com uma condição de guarda, para cada serviço modelado, no formato: `[(actual=id_do_componente) andalso (serv=serviço_do_componente)]`, sendo *id_do_componente* o identificador do componente modelado e *serviço_do_componente* o nome do serviço modelado. Como cada componente possui um identificador único e nenhum componente implementa dois ou mais serviços com o mesmo nome [Almeida et al. 2004b], apenas uma transição poderá disparar. O disparo de uma transição levará a ficha para o modelo do componente que expresse o comportamento do serviço requisitado. As variáveis *actual* e *serv* são atributos da ficha de requisição.

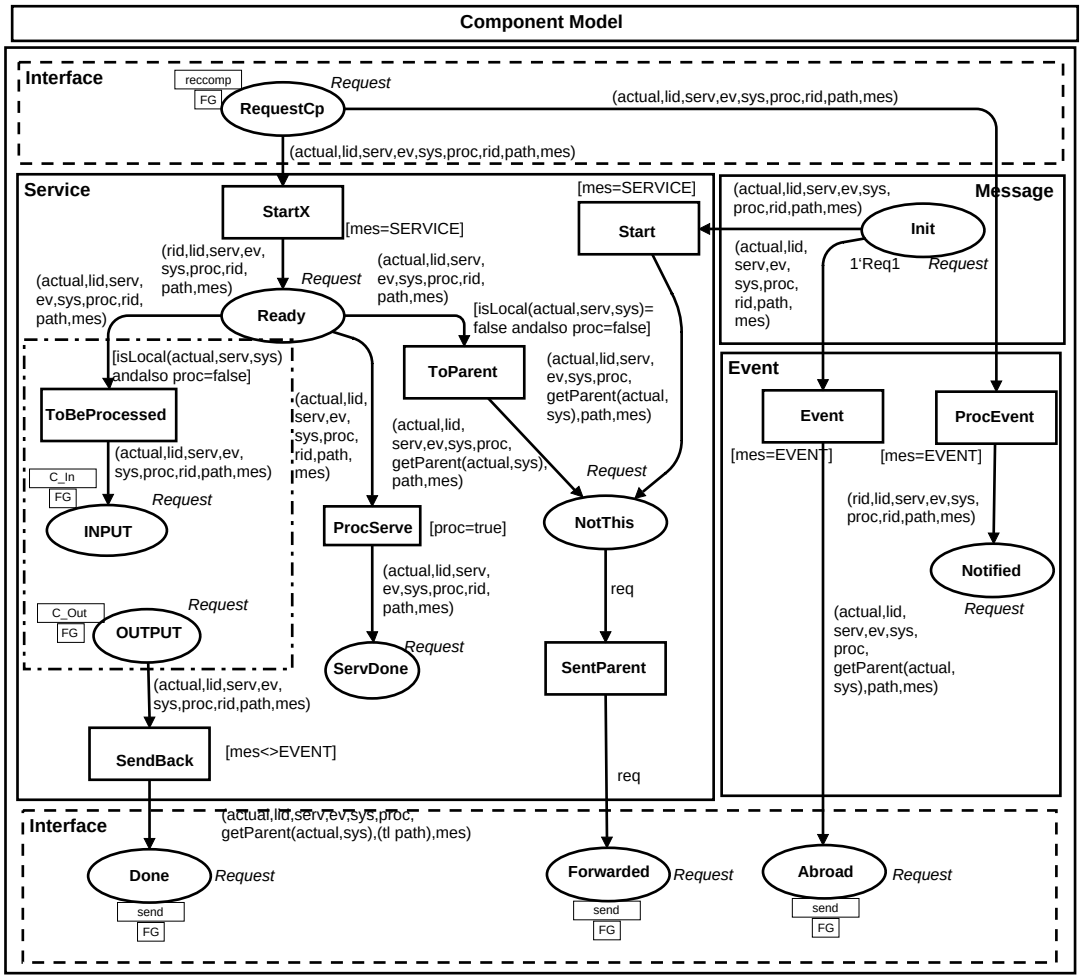

Figure 1. Modelo do arcabouço COMPOR-CPN

4. Modelagem e Verificação do Arcabouço ArCo

A seguir, descreve-se a aplicação do arcabouço COMPOR-CPN à modelagem e verificação de um sistema de informação construído de sobre o ambiente para gerência de comunidades virtuais denominado ArCo [Almeida et al. 2004a]. ArCo dispõe de ferramentas para a interação e colaboração de atores, componentes de infra-estrutura e interface gráfica, além de suportar integração com outros sistemas. ArCo é baseado no padrão arquitetural de camadas funcionais, sendo definidos componentes para as camadas de infra-estrutura, camada de núcleo, ferramentas e interface gráfica unificada [Almeida et al. 2004a].

O foco deste artigo é a modelagem e a verificação da camada de ferramentas. Na camada de ferramentas estão definidas as ferramentas de colaboração e gerenciamento construídas a partir de componentes e ligadas às comunidades. As ferramentas são encapsuladas em componentes. Em sua versão atual, são definidos componentes de *chat*, fórum, e-mail, *whiteboard* e video conferência [Almeida et al. 2004a].

Definição da arquitetura

O primeiro passo para a modelagem e verificação do sistema de informação é a definição de sua arquitetura nos termos da CMS. Como descrito anteriormente, existem duas entidades básicas na CMS: contêineres e componentes. Sendo assim, basta

definir qual a hierarquia de componentes e contêineres do sistema de informação, de acordo com a divisão em módulos da arquitetura. Por exemplo, um sistema seguindo a arquitetura clássica de três camadas poderia ser definido como um contêiner raiz (*Sistema*), composto por outros três contêineres (*Apresentação*, *Negócio* e *Armazenamento*), os quais seriam compostos pelos componentes que implementam as funcionalidades da aplicação [Almeida et al. 2004b].

A arquitetura definida deverá então ser refletida na estrutura do arcabouço COMPOR-CPN para que possa ser verificada. Uma vez que o arcabouço é independente de uma arquitetura específica de sistema de informação, a definição da arquitetura para o estudo de caso deve ser feita de forma não intrusiva na rede - como uma variável global.

Portanto, para determinar a arquitetura de um sistema específico no arcabouço COMPOR-CPN, basta definir o valor de uma variável denominada *System*. O formato desta variável é uma lista de componentes e contêineres, que determinam a hierarquia da arquitetura, seguindo o formato: `(nome,[servicos],pai,[filhos],[eventos])`. O contêiner cujo valor de *pai* for igual ao seu *nome* será considerado o raiz.

No contexto do ArCo, o seguinte valor foi definido para *System*, referente ao módulo de ferramentas [Almeida et al. 2004a]. A partir do valor desta variável, a simulação do modelo do COMPOR-CPN, assim como sua verificação, torna-se específica para o ArCo. A linguagem de definição da variável é CPN/ML [Jensen 1997].

```
val System = [(Forum, [PostMsg], Tools, [], [NewMsg]),
    (Email,[SendMail], Tools, [], [MailSent]),
    (Whiteboard, [Paint,Clean], Tools, [], [OnGoOut]),
    (VideoConf, [VideoStream], Tools,[], []),
    (Chat, [DisconnectChat, ConnectChat], Tools, [],[OnEnter,OnGoOut]),
    (Tools, [DisconnectChat, ConnectChat], Tools,[Chat, Interface], [OnEnter,OnGoOut])
    ]
```

Modelagem dos componentes

A modelagem de cada componente do sistema é realizada de forma separada. Cada componente deverá possuir seus lugares de entrada e saída, além de sua transição com condição de guarda referente a cada um de seus serviços. Isto permite que os componentes sejam independentes uns dos outros e da estrutura do sistema.

Um dos componentes de comunicação do ArCo será utilizado como exemplo da modelagem de componentes - o componente *chat*. Serão detalhados dois serviços deste componente: *entrar* e *sair*. Neste último, haverá a possibilidade de sair de apenas uma sala ou de todas de uma só vez. Cada usuário só poderá enviar mensagens às salas de *chat* na qual ele esteja presente. Além disso, as mensagens que chegarem numa sala num determinado momento, só serão visíveis aos usuários que estiverem presentes naquele momento.

O modelo CPN do *chat*, ilustrado na Figura 2, modela todas as especificações mencionadas. O modelo tem dois lugares de fusão, para entrada e saída da ficha de requisição. Além disso, o lugar de fusão de entrada possui arcos de saída que levam a duas transições. Estas transições possuem guardas que verificam qual o serviço requisitado. De acordo com o serviço, a ficha pode seguir caminhos diferentes pelo modelo.

Ao receber uma requisição *ConnectChat*, uma ficha do tipo *user* é adicionada ao lugar *Connected*. Neste lugar estarão todos os usuários conectados. Após estar conectado, o usuário poderá ou não entrar numa sala de *chat*. A transição *Enter_Chat* garante, através de sua guarda, que um usuário só entrará em salas nas quais ele ainda não esteja. No lugar

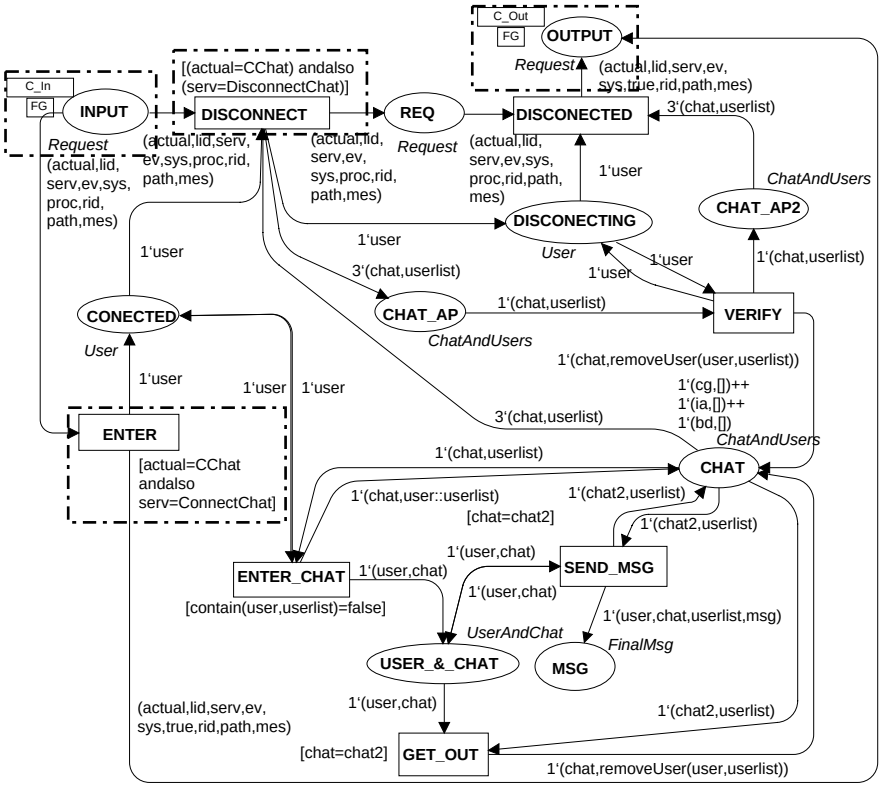

Figure 2. Modelagem do componente de chat.

User_&_Chat se encontra uma tabela de usuários por salas. A partir de então, o usuário poderá enviar mensagens ou poderá sair de alguma sala na qual se encontre.

Caso receba uma requisição *DisconnectChat*, a rede irá retirar o usuário de todas as salas nas quais ele se encontre. Para isso, a transição *DISCONNECT* transfere todas as fichas do lugar *CHAT* para o lugar *CHAT_AP* e adiciona uma ficha, representando o usuário que deseja sair, no lugar *DISCONNECTING*. Para cada ficha presente em *CHAT_AP*, a transição *VERIFY*, enviará uma ficha idêntica para *CHAT_AP2* e, através do arco de saída que leva ao lugar *CHAT*, verifica se o usuário está na sala para então removê-lo. Caso não esteja na sala, a expressão deste arco apenas coloca a ficha no lugar *CHAT*. O peso do arco de saída do lugar *CHAT_AP2* garante que a transição *DISCONNECTED* só estará habilitada quando todas as salas tiverem sido verificadas.

Foram definidos cenários de simulação usando os cinco componentes declarados na variável $System$: requisições para postar mensagem em *fórum*; para enviar *e-mail*; transferência de *vídeo*; operações de escrita e leitura no *whiteboard*; e requisições para entrar e sair de salas de *chat*. Em todos os cenários, as propriedades especificadas foram garantidas. Entretanto, as simulações não cobrem todos os possíveis comportamentos do modelo, podendo haver algum cenário em que o comportamento do modelo não esteja de acordo com a especificação do sistema. Para evitar tal situação, faz-se necessário o uso da técnica de verificação de modelos.

Verificação do modelo

A verificação de modelos é uma técnica que confronta especificações de propriedades do sistema modelado com o espaço de estados do modelo (grafo dirigido onde cada nó representa um possível estado do modelo). Estas especificações são expressas

em lógica temporal. Caso uma propriedade falhe em um determinado estado, um caminho, do estado inicial ao estado de falha, é apresentado como contra-exemplo para tal propriedade. Desta forma, verifica-se se existe um cenário em que o sistema não funcione corretamente, considerando todos os cenários possíveis.

Neste trabalho, o espaço de estados (também conhecido como grafo de ocorrência) é gerado pelo conjunto de ferramentas Design/CPN, e as propriedades são escritas usando a biblioteca ASK/CTL [Christensen and Mortensen 1996]. A partir do modelo do estudo de caso, foram gerados 228026 estados como grafo completo. As seguintes propriedades foram verificadas: (i) sempre que alguém *requisitar e obtiver conexão* (proposição *pa*) num *chat*, receberá uma resposta que foi *conectado* (proposição *pb*). A fórmula em lógica temporal que descreve esta propriedade é: AG(pa $\rightarrow$ AF(pb)); (ii) toda mensagem enviada a uma sala num determinado instante, só será recebida pelos usuários que estiverem naquela sala e naquele instante. Proposições: *mensagem enviada a sala X* (*pa*), *usuários que estão na sala X recebem mensagem* (*pb*), *usuários que não estão na sala X não recebem mensagem* (*pc*). Fórmula: AG(pa $\rightarrow$ (AF(pb) $\wedge$ AG($\neg$pc)); (iii) e caso uma *requisição para desconectar* (proposição *pa*) seja feita, o usuário será *retirado de todas as salas* (proposição *pb*) e será *desconectado* (proposição *pc*). Fórmula que descreve esta propriedade: AG(pa $\rightarrow$ AF(pb) $\wedge$ AF(pc)). Além disso, as propriedades referentes à interação entre os componentes permanecem válidas, não sendo encontrados cenários com *deadlocks* ou *livelocks*.

5. Trabalhos relacionados

Em [Bernd Finkbeiner and Ingolf Krüger 2001], é apresentada uma investigação sobre o uso de *Message Sequence Charts* (MSCs) como uma técnica de especificação para a composição de sistemas a partir de componentes. Neste trabalho, é definida uma regra de prova de decomposição em MSCs e é sugerido um operador de composição para especificações MSC de tais componentes e descreve diferenças para os operadores usados pela composição de cenários. Em [Silva and Perkusich 2005], é introduzido um processo para a modelagem e verificação formal de sistemas de software baseados em componentes. Tal processo é baseado em arquitetura de software, componentes, e reúso de modelos de Redes de Petri Coloridas. O trabalho apresentado neste artigo é complementar ao processo proposto em [Silva and Perkusich 2005], o qual é mais voltado para ao processo de modelagem. Em um contexto mais pragmático, em [Heineman et al. 2005] foi realizado um estudo de caso para validar a eficiência de um projeto baseado em componentes de um servidor *web multi-threaded*. Em [Aniorte 2003] é proposto um modelo de componentes para desenvolvimento de *software* adaptável. A interface do modelo de componentes é descrita na forma de pontos de interação. Esses pontos de interação são usados para gerenciar diferentes tipos de interações, visando construir um grafo de interações permitindo a integração de componentes reutilizados.

6. Conclusões

Neste artigo apresentou-se a utilização de um arcabouço denominado COMPOR-CPN para a modelagem e verificação formal de sistemas de informação baseados em componentes. O arcabouço representa a modelagem em redes de Petri coloridas da especificação de componentes COMPOR e é independente da arquitetura do sistema a ser modelado e verificado. A ferramenta Design/CPN é utilizada para modelagem dos componentes e geração do espaço de estados. A verificação do modelo é realizada através da biblioteca ASK/CTL, integrada ao Design/CPN.

Como estudo de caso, um sistema de informação construído sobre um software para a gerência de comunidades virtuais foi modelado e verificado utilizando o

COMPOR-CPN. Durante esta verificação, um dos problemas encontrados foi a explosão do espaço de estados [Valmari 1998]. Isso ocorreu devido à presença de muitos componentes integrados na modelagem, o que causou o aumento no número de estados gerado. Sendo assim, para modelar sistemas com muitos componentes, faz-se necessário verificar conjuntos de componentes separadamente ou, caso seja imprescindível verificar a interação entre todos de uma só vez, empregar uma maior abstração aos modelos internos dos componentes.

Como trabalho futuro, pretende-se utilizar o arcabouço para verificar sistemas de informação de larga escala, cujos modelos de componentes são complexos e concorrentes. A viabilidade da utilização deste arcabouço para tais domínios seria de grande importância para unir sistematização e simplicidade à inerente necessidade de confiança no funcionamento de tais sistemas.

References

Almeida, H. O., de Barros Costa, E., Barbosa, N. M., Bublitz, F. M., and de Andrade Barbosa, A. (2004a). Um Arcabouço de Software Livre baseado em Componentes para a Construção de Ambientes de Comunidades Virtuais de Aprendizagem na Web. In *Anais do XV Simpósio Brasileiro de Informática na Educação - SBIE'05*, pages 188–196, Manaus, Amazonas, Brasil.

Almeida, H. O., Perkusich, A., Paes, R. B., and Costa, E. B. (2004b). Composição Dinâmica de Componentes para Aplicações com Mudanças Freqüentes de Requisitos. In *Anais do 4o Workshop de Desenvolvimento Baseado em Componentes*, pages 9–14, João Pessoa, Paraíba, Brasil.

Aniorte, P. (2003). Componet-Based Software Development: From Component Model to Software Architecture. In *ICSR7 2002 Workshop on Component-based Software Development Processes*, Austin, Texas, USA.

Bernd Finkbeiner and Ingolf Krüger (2001). Using Message Sequence Charts for Component-Based Formal Verification. In Giannakopoulou, D., Leavens, G. T., and Sitaraman, M., editors, *SAVCBS 2001 Proceedings: Specification and Verification Component-Based Systems*, pages 32–41. ISU TR #01-09.

Christensen, S. and Mortensen, K. H. (1996). *Design/CPN ASK-CTL Manual*. University of Aarhus.

de Almeida, H. O., da Silva, L. D., da Silva Oliveira, E., and Perkusich, A. (2005, to appear). A Formal Approach for Component Based Embedded Software Modelling and Analysis. In *Proceedings of IEEE International Symposium on Industrial Electronics - ISIE'05*, Croatia.

Heineman, G. T., Crnkovic, I., Schmidt, H. W., Stafford, J. A., Szyperski, C. A., and Wallnau, K. C., editors (2005). *Component-Based Software Engineering, 8th International Symposium, CBSE 2005, St. Louis, MO, USA, May 14-15, 2005, Proceedings*, volume 3489 of *Lecture Notes in Computer Science*. Springer.

Jensen, K. (1997). *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, volume 2. Springer-Verlag.

Silva, L. D. and Perkusich, A. (2005). Composition of Software Artifacts Modelled Using Colored Petri Nets. *Science of Computer Programming*, 56(1):171–189.

Valmari, A. (1998). The State Explosion Problem. In Reisig, W. and Rozenberg, G., editors, *Lectures on Petri Nets 1: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 429–528. Springer-Verlag.