Redes de Petri Colorida no Desenvolvimento de Objetos de Aprendizagem: Uma Análise das Propriedades Comportamentais do Modelo LOCPN

Francisco Herbert Lima Vasconcelos, Lucas Lopes do Amaral, Maria de Fátima Costa de Souza, José Aires de Castro Filho, Mauro Cavalcante Pequeno, Giovanni Cordeiro Barroso[1]

[1]Universidade Federal do Ceará.
Campus do Pici, - Bloco 901 - 1º Andar, CEP: 60455-760 – Fortaleza-CE - Brasil

herbert@virtual.ufc.br, lopesdoamaral@gmail.com, gcb@fisica.ufc.br

Abstract. *This paper aims to make an analysis on the Petri Nets (RdP) behavioral properties focusing at its semantics on Learning Objects development. We use the Learning Object Colored Petri Nets, also known as LOCPN, an extension of the traditional Colored Petri Nets (RdPC), to observe those properties and see if it go towards the initial proposal. The reached results demonstrate the viability of the use of the properties behavior of RdPC in the development of OA.*

Resumo. *Este trabalho tem como principal objetivo à análise das propriedades comportamentais das Redes de Petri (RdP) com foco em sua semântica para o desenvolvimento de Objetos de Aprendizagem. Neste artigo realizou-se um experimento de validação com o modelo LOCPN, uma extensão das já tradicionais Redes de Petri Coloridas (RdPC), utilizadas para construir e observar estas propriedades e avaliar se vão ao encontro das definições iniciais. Os resultados alcançados demonstram a viabilidade do uso das propriedades comportamentais das RdPC no desenvolvimento de OA.*

1. Introdução

A utilização de recursos tecnológicos aplicados na educação objetivando melhorias no processo ensino-aprendizagem tem crescido consideravelmente desde a década de 40 com as máquinas de ensinar de B. F. Skinner, um dos defensores da linha comportamentalista. Desde então, com o desenvolvimento das tecnologias digitais da informação e comunicação (TEDIC), estes recursos outrora mecânicos e de poucas possibilidades ganharam espaço entre o mundo cibernético dos computadores com o desenvolvimento de softwares educativos.

Hoje temos um conceito paralelo ao de Softwares Educativos (SE) que é o de Objeto de Aprendizagem (OA), que de tão largo uso atualmente é definido por [Wiley, 2007] e [IEEE, 2007] como "*qualquer entidade digital ou não, que pode ser usada, reusada ou referenciada durante um processo de aprendizagem suportado pela tecnologia*".

Os OA tem por característica sua granularidade, isto é, diferente de SE que abordam um grande número de conteúdos de uma mesma área, os OA são objetivos, atingem uma única temática, tornando-se mais adaptável e reutilizável em diversas disciplinas escolares.

O fato de serem atômicos não faz os OA serem de mais simples desenvolvimento que os demais sistemas de informação empregados em vários setores produtivos. Eles precisam traduzir com fidelidade o que é demandado por professores e alunos, não podendo-se deixar espaços para dubiedades que podem gerar frustração dos usuários.

Nesse contexto, surgem disciplinas oriundas da Engenharia de Softwares tradicional que focam exclusivamente no processo de desenvolvimento de aplicações voltadas para educação, como a Engenharia de Softwares Educativos (ESE), que visa a criação, adaptação e aplicação de métodos novos ou tradicionais para a melhoria das aplicações educacionais.

Aprofundaremos o tema do desenvolvimento de softwares na seção 2 deste trabalho, seguindo com a fundamentação teórica da engenharia de software educativo e da engenharia de requisitos na seção 3. Nas seções 4 e 5, apresentaremos o LOCPN e uma aplicação deste modelo em um OA. Traremos as discussões sobre as propriedades comportamentais e suas aplicações no desenvolvimento dos OA na seção 6, concluindo com a seção 7, com breves considerações finais sobre o trabalho e trabalhos futuros.

2. Técnicas de Desenvolvimento de Software

Segundo [Sommerville, 2003], software é um conjunto de instruções lógicas, desenvolvidas em linguagem específica, que permite ao computador realizar as mais variadas tarefas do dia-a-dia de empresas, profissionais de diversas áreas e usuários em geral. Esta definição nos mostra bem quão genérico ou intangível [Sommerville, 2003] é o software, que ainda segundo [Sommerville, 2003] é irrepetível posto que é desenvolvido para pessoas diferentes ainda que sejam da mesma área de aplicação.

Disciplinas como a Engenharia de *Software* criam metodologias que ajudam no desenvolvimento destes produtos. Modelos de processos, que definem um conjunto de tarefas sistemáticas implementadas para se obter um *software*, padrões de interações entre profissionais de áreas diferentes, técnicas de descrição formal, como será abordado neste trabalho, entre tantas outras metodologias surgem de acordo com o domínio a ser modelado em um *software*.

Não acontece diferente com os SE ou OA, também para estes tem-se desenvolvidos técnicas como em [Amaral, 2006], [Monteiro, 2006] e [Gomes, 2003] que visam, indiferente ou não a uma determinada linha pedagógica, a melhor captação de requisitos especificados pelos clientes-professores ou ainda clientes-alunos e a transposição destas informações para o software.

Atualmente, grande parte dos OA produzidos são especificados com auxílio de um roteiro em forma de *storyboard* e linguagem natural, metodologia suscetível a falhas, incompletudes de informações e ambigüidades como trata [Sommerville, 2003] e Souza, 2007]. Desta maneira buscam-se técnicas que aprimoram a captação das exigências ou requisitos como a já tradicional modelagem UML [Booch, 2005], que apesar de modelar, não viabiliza a detecção de falhas ou inconsistências ainda nas fases iniciais da análise de requisitos, isto é, com UML não podemos verificar e validar requisitos com facilidade.

Para essa atividade tem-se recorrido a Técnicas de Descrição Formal (TDF), como Redes de Petri, SDL ou LOTOS, e ainda técnicas semi-formais como os Use Case Maps [Souza, 2007]. Estas técnicas têm o poder de evidenciar ainda no princípio

potenciais falhas que acarretariam em prejuízos, seja através de seus recursos gráficos, seja pela robustez da linguagem.

Além disso, tais técnicas, no desenvolvimento de OA, devem captar, de acordo com [Prata, 2007], características que são importantes em uma interface: a condução, a afetividade, a consistência, o significado de códigos e denominações e gestão de erros.

Uma interface bem elaborada permite ao usuário utilizá-la com facilidade, e é um grande desafio conciliar usabilidade e design. Para tanto, é necessário fazer com que ela seja a mais compreensível possível, e adequada ao público-alvo, devendo-se evitar uma sobrecarga de informações. Um bom desenvolvimento de um software educativo, deve passar pelas recomendações mínimas da engenharia de software educativo (ESE). Na seção que se segue iremos expor os fundamentos teóricos iniciais da ESE.

3. A Engenharia de Software Educativo e a Engenharia de Requisitos

3.1 Aspectos Fundamentais da Engenharia de Software Educativo

A engenharia de software educativo (ESE) segue as fases genéricas de um processo sistemático de desenvolvimento de software. A ESE leva em consideração os seguintes aspectos: a solidez da análise, como ponto de partida para analisar problemas em função de necessidades educacionais; a definição de objetivos didático-pedagógicos; a decomposição dos objetos definidos, para obter a estrutura das tarefas e de possíveis atividades de aprendizagem; o design educativo que deve ser projetado em função das atividades de aprendizagem [Galvis, 1992]. Portanto, a ESE é uma disciplina em constante evolução que possibilita aplicar idéias que visam fazer com que o aprendiz possa interagir com ambientes educativos informatizados que acrescentem valores aos meios educativos que estão disponíveis, normalmente, para auxiliar no processo de ensino-aprendizagem. A metodologia de modelagem a partir de Redes de Petri (RdP) que pretende-se utilizar neste trabalho para o desenvolvimento de Objetos de Aprendizagem (OA), segue as mesmas etapas de um processo sistemático para o desenvolvimento de software convencional (análise, definição dos requisitos, design, desenvolvimento, validação e verificação). Entretanto, dar-se ênfase particular aos seguintes aspectos: solidez da análise de requisitos como ponto de partida; a inclusão de teorias de aprendizagem cognitivistas e a modelagem aplicada ao ensino, a avaliação pedagógica do material, a documentação adequada e suficiente de tudo que se realiza em cada etapa.

Para contextualizar a evolução de todos os processos da metodologia, Galvis [1992] concebeu um modelo sistemático que permite visualizar os ciclos tanto para seleção quanto para desenvolvimento de software educativo (SE), conforme mostra a Figura 1.

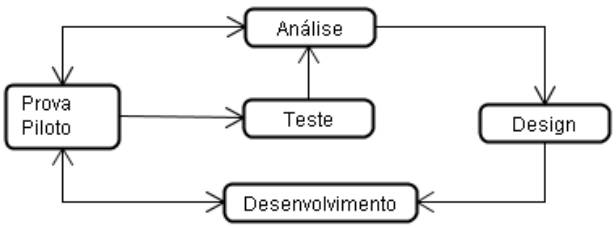

Figura 1. Modelo Sistemático para Seleção e Desenvolvimento de Software.

O ciclo de desenvolvimento do SE inicia com a análise de necessidades educativas, para proceder com as demais etapas que orientam o design educativo. A finalidade da análise de necessidades educativas é encontrar alternativas para solucionar problemas existentes em um determinado contexto educacional. Nesta etapa, normalmente são analisadas as situações problemáticas existentes, suas causas e possíveis soluções, para então, determinar quais são as aplicações que podem gerar melhores resultados de aprendizagem. Estas necessidades podem ser compreendidas como a definição dos requisitos do sistema. Desta forma faz-se necessário compreendermos o que são tais requisitos e como iremos utilizar as técnicas formais das Redes de Petri Colorida afim de determiná-los.

3.2 Engenharia de Requisitos

De modo geral, os requisitos representam as necessidades do sistema de software e as restrições impostas a ele. Um requisito é neste contexto uma condição ou capacidade necessária para um usuário resolver um problema ou alcançar um objetivo, definindo assim o domínio do problema. Mas a engenharia de requisitos tem um escopo mais abrangente, pois envolve o universo de informações que contextualizam o sistema e todos os *stakeholders* [Sommerville, 2003]. Entretanto, num dos primeiros trabalhos realizados na área, Bell e Thayer [1976] observaram que muitos requisitos são inadequados, inconsistentes, incompletos e ambíguos e que eles tem um grande impacto na qualidade do software final. A partir dessa observação eles concluíram que requisitos para um dado sistema não podem ser levantados naturalmente, ao contrário eles precisam ser projetados e necessitam de contínuas revisões.

Desta forma para compreendermos a Engenharia de Requisitos é fundamental entendermos exatamente o que são requisitos de um sistema. Sendo assim, podemos descrever requisitos como as funções que deverão ser incorporadas pelo software, quando inserido em seu contexto de funcionamento. No entanto, necessidades como desempenho, integridade, disponibilidade e segurança seriam difíceis de acomodar nessa definição preliminar de requisitos. Apesar de tais qualidades transformarem-se, em algum nível de abstração, em funcionalidades do novo sistema, no nível de abstração no qual requisitos estão sendo tratados, tais qualidades não são funcionalidades do sistema em especificação. Dessa forma, segundo Sommerville [2003], podemos classificar os requisitos de um software da seguinte forma:

- **Requisitos Funcionais:** são declarações de funções que o sistema deve fornecer, como o sistema deve agir a entradas específicas e como deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais podem também explicitamente declarar o que o sistema não deve fazer;
- **Requisitos Não-Funcionais:** são restrições sobre os serviços ou as funções oferecidos pelo sistema. Entre eles, destacam-se restrições sobre o processo de desenvolvimento, padrões, entre outros. Na Figura 2 é definida uma classificação para os requisitos não funcionais;
- **Requisitos de Domínio:** são requisitos que definem funções específicas de determinados domínios de aplicação. Esses requisitos tanto podem ser funcionais como não funcionais.

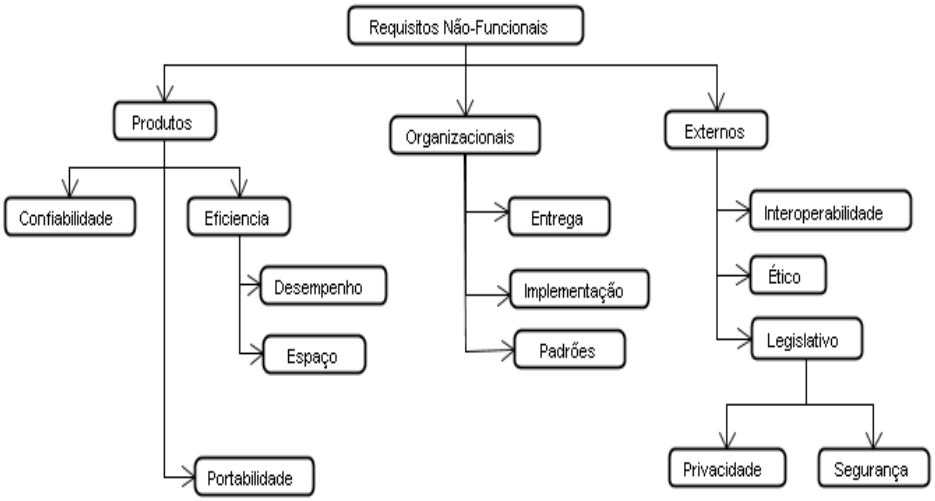

Figura 2. Requisitos Não-Funcionais.

A engenharia de requisitos é uma etapa essencial do processo de desenvolvimento de sistemas de software, que compreende uma definição completa do comportamento externo desse sistema, tanto em termos de requisitos funcionais quanto de requisitos não funcionais. Vários estudos têm constatado que a definição inadequada de requisitos é responsável por parte significativa dos erros no software, cuja eliminação torna-se cada vez mais difícil e dispendiosa à medida que o processo de desenvolvimento do software avança para as etapas subseqüentes de projeto e implementação [Kirner, 1996].

A engenharia de requisitos compreende fases distintas, porém inter-relacionadas, de elicitação, especificação e validação [Castro, 1995], [Kirner, 1996]. Tais fases devem ser definidas de forma clara e precisa, para que durante o desenvolvimento de um sistema se garanta sua viabilidade em termos de funcionamento e especificação.

Uma definição genérica para a Engenharia de Requisito é que ela é uma atividade que objetiva estabelecer o que o cliente requer de um sistema de software. Segundo o IEEE [IEEE, 1984], engenharia de requisitos corresponde ao processo de aquisição, refinamento e verificação das necessidades do cliente para um sistema de software, objetivando-se ter uma especificação completa e correta dos requisitos de software.

Portanto, diante do que foi exposto, neste trabalho pretende-se agregar a especificação de requisitos de OA a partir da análise das propriedades comportamentais das Redes de Petri.

4. O Modelo LOCPN

O modelo LOCPN foi proposto por [Souza, 2007] e tem como característica principal o uso das Redes de Petri Coloridas na Produção de Objetos de Aprendizagem. Neste artigo realizamos uma aplicação deste modelo por meio de um estudo de caso com um objeto de aprendizagem. Antes de iniciarmos esta análise, iremos expor os fundamentos teóricos iniciais das Redes de Petri Colorida e as características do Modelo LOCPN.

4.1 Redes de Petri Coloridas

O principal objetivo das RdPC [Jensen, 1997] é a redução do tamanho do modelo, permitindo que fichas individualizadas (coloridas) representem diferentes processos ou recursos em uma mesma sub-rede. Nas RdPC as fichas são representadas por estruturas de dados complexas. Deste modo, as fichas podem conter informações. Além disso, cada lugar armazena fichas de um certo tipo definido e arcos realizam operações sobre elas. As transições determinam a dinâmica da RdPC e podem apresentar "expressões de guarda". Estas, por sua vez, indicam os tipos de fichas que possibilitam ativar uma transição.

Uma RdPC é composta por três partes: estrutura, inscrições e declarações. A estrutura é um grafo direcionado, com dois tipos de nós (lugares e transições), com arcos interconectando nós de tipos diferentes. As inscrições são associadas aos lugares, transições e arcos. As declarações são tipos, funções, operações e variáveis. Quando a expressão do arco é avaliada, ela gera um multi-conjunto de fichas coloridos. Expressões podem conter constantes, variáveis, funções e operações definidas nas declarações, e não produzem efeito colateral.

Para [Souza, 2007] as redes de Petri coloridas possuem uma grande quantidade de recursos que podem ser utilizados para especificar detalhadamente as ações de um sistema, permitindo a formalização na sua produção.

4.2 Caracterização do Modelo LOCPN

Na realização da modelagem do OA que será apresentado neste artigo, adotamos o Modelo LOCPN, proposto por [Souza, 2007]. Este modelo é melhor compreendido, a partir de suas definições e características destacadas abaixo:

Definição 1. Uma *Learning Objects production with Colored Petri Net* é definida como: LOCPN = (S, SN, AS, PN, PT, PA, AT), onde:

S é um conjunto finito de páginas, onde cada i E S é uma RdPC (Σ*i, Li, Ti, Ai, Ni, Ci, Gi, Ei, Ii*), com as seguintes características:

i. Σ*i* = <*C0, e*> representando um conjunto não vazio de fichas (cores), no qual *C0* representa cores ordinárias que não carregam nenhuma informação e *e*, quando presente em um lugar que represente uma tela/janela de um OA, simboliza a ativação da ela/janela.
ii. *Li = {l1,l2, ...,lm} U {t1, t2, ..., tn}* é um conjunto finito de lugares, que é formado por dois subconjuntos, o primeiro, representa locais comuns (simbolizado graficamente por círculos/ elipses com bordas simples), o segundo é formado por lugares que representam telas/ janelas de um OA (simbolizado graficamente por círculos/elipses com bordas grossas).
iii. SN é um conjunto de nós de substituição, onde SN$\underline{C}$T.
iv. SA é uma função de atribuição de páginas;
v. PN é um conjunto de nós que representam portas
vi. PT é uma função de tipo de porta, definida como PT: PN →{in, ou, in/out}
vii. PA é uma função de atribuição de portas;
viii. AT é uma função de ativação, que representa a ativação da tela de um OA. Essa função é uma extensão da função Ci e é definida por AT: Lt → {e}, em que Lt é o subconjunto de lugares que representa telas/janelas de um OA.

A partir das definições iniciais propostas por [Souza, 2007], podemos apresentar ainda as características do LOCPN para especificar o fluxo de ações em um OA. Segundo [Souza, 2007], temos algumas situações que são possíveis na modelagem de fluxo entre as telas de um OA, ou seja, existe um conjunto de transições entre telas passíveis de serem modeladas em um OA. De um modo geral esse conjunto pode ser reduzido às seguintes situações:

- *Fluxo Direto Interativo:* Uma tela á apresentada a partir da realização de alguma atividade de interação do usuário, como por exemplo, um botão explicitamente invoca uma tela com informações adicionais sobre o conteúdo que está sendo apresentado no OA.
- *Fluxo Direto Automático:* Uma tela é apresentada espontaneamente sem a requisição direta do usuário. Nesse caso não existe uma invocação explicita do usuário. Um exemplo para essa situação é a apresentação de um *pop-up* com algum aviso sobre a utilização de um exercício no OA.
- *Fluxo Direto Condicional:* Nesse caso uma tela deve ser apresentada dependendo de alguma informação relacionada ao estado da tela que realizou a invocação. Por exemplo, uma tela pode ser apresentada caso o usuário do OA tenha errado três vezes consecutivas um determinado exercício.

5. Estudo de Caso

Para a verificação das potencialidades do uso das RdPC no desenvolvimentos de OA realizamos um estudo de caso da aplicabilidade do LOCPN. Partindo das técnicas já mencionadas anteriormente, desenvolvemos um protótipo em alto nível que demonstrasse a aplicação desses conceitos e, logo após a implementação do modelo, avaliamos a metodologia e os resultados.

O modelo desenvolvido simula uma aplicação básica onde o usuário é capaz de, após uma tela inicial, desenvolver uma atividade genérica que aqui é representada por um lugar que representa uma janela ou tela, como previsto pelo LOCPN, sem, contudo, ir afundo nesta atividade. Em qualquer das telas já citadas o usuário também pode solicitar a tela de ajuda, para orientar-se quanto ao objetivo daquela atividade. Esta dinâmica está expressa na Figura 3.

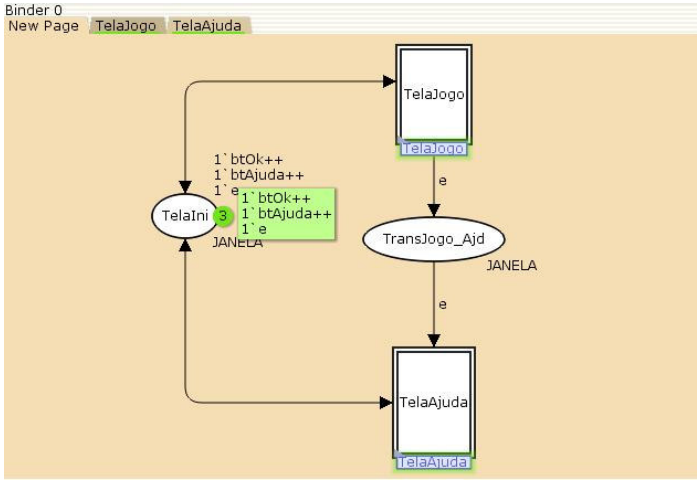

Figura 3. Estrutura Principal do Sistema.

Observamos (Figura 3) a presença da ficha “e”, responsável pela atribuição da apresentação da tela em determinado tempo, bem como outras fichas que representam outros elementos de interface responsáveis pela determinação do fluxo das telas como o “btOk”, que, juntamente com a ficha de ativação de janela “e”, leva do lugar “TelaIni”, para a o lugar “jogar” como expresso na Figura 4.

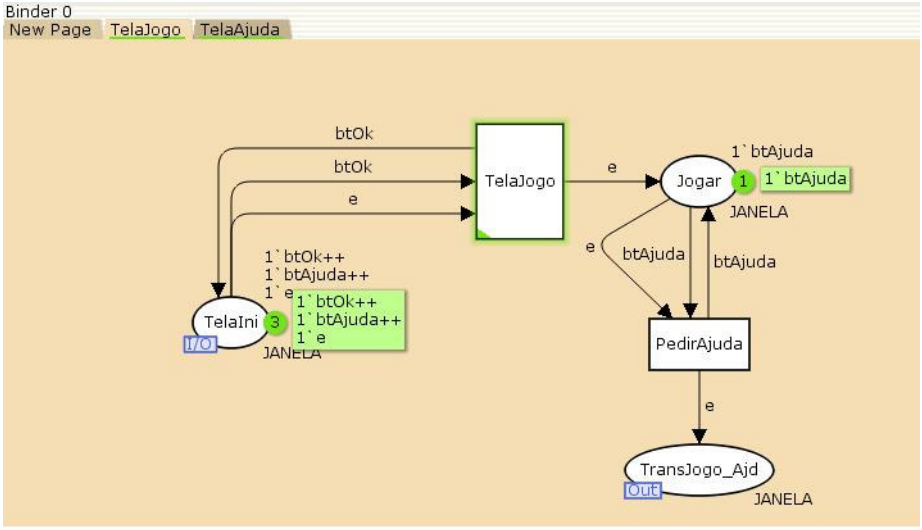

Figura 4. Rede que Representa a Atividade.

Para elevar ainda que em um grau reduzido a complexidade do estudo de caso, criamos a janela de ajuda, que terá um fluxo análogo ao mostrado anteriormente como mostra a Figura 5 abaixo.

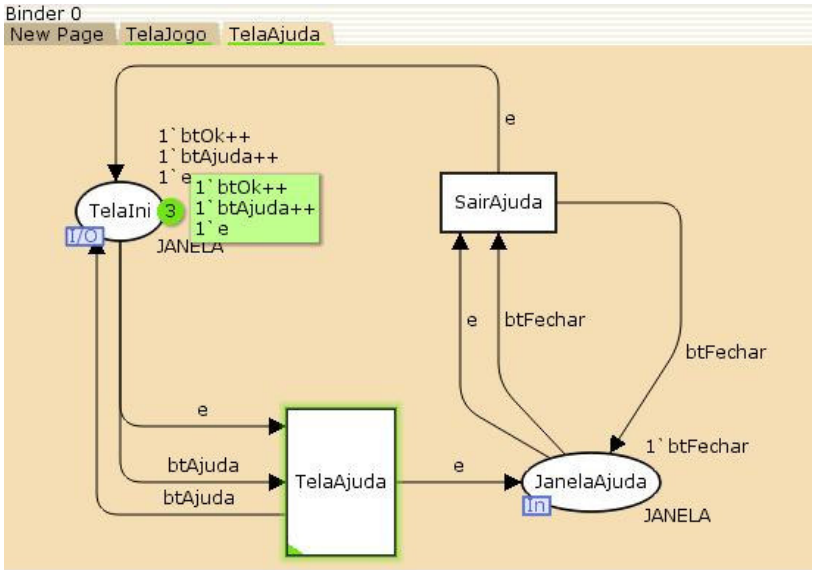

Figura 5. Rede que modela a janela de ajuda.

6. Resultados e Contribuições das Propriedades das RdPC

O estudo realizado na seção anterior permite destacar como principais contribuições verificadas a partir das propriedades das RdPC no desenvolvimento de OA, os seguintes requisitos funcionais e não-funcionais:

(a) Alcançabilidade: ao verificarmos no modelo a propriedade comportamental de alcançabilidade, percebe-se a capacidade de todos os *lugares* (que representam as telas) serem alcançados, ou seja, não teremos telas desenvolvidas que serão impedidas por algum fluxo de serem visualizadas pelo usuário.

(b) Limitação: a limitação pode nos deixar cientes de recursos que poderão ou não ser ilimitados, como o número de usuários simultâneos ou a quantidade de vezes que pode ser executada alguma tarefa ou até mesmo qualquer outra quantidade de recurso que possa ser disponibilizado no OA. Ressalta-se a possibilidade de uma adequação ao modelo cognitivo que se está aplicando no desenvolvimento deste OA.

(c) Vivacidade: esta propriedade relaciona-se com o desenvolvimento do OA, posto que garante uma não frustração no uso do mesmo por uma falha de fluxo das janelas. O que garante o desempenho das atividades desejadas por aqueles que o planejam.

(d) Reversibilidade e Estado de Passagem: em uma interface como a de um OA, é desejável a possibilidade de navegação e exploração, conhecida como *hands-on*, na qual o usuário torna-se livre para navegar por todo o ambiente livremente, explorando todas as suas possibilidades, bem como desfazer ações.

(e) Persistência: esta propriedade trata do disparo de uma transição que desabilita ou não outras transições. Neste contexto abordado, tem-se um duplo significado: o primeiro faz referência ao disparo de uma transição interna a uma rede que modela uma janela, neste nível, várias transições podem estar habilitadas simultaneamente e sendo disparadas sem desabilitar as demais. Em outro nível, quando se refere ao fluxo entre janelas, o disparo de transição pode habilitar somente uma janela, não podendo outro disparo de passagem entre janelas ser efetuado.

7. Considerações Finais

A partir da realização deste experimento podemos observar a viabilidade do desenvolvimento de OA com o uso da técnica LOCPN. Observa-se essa viabilidade tanto em sua fundamentação quanto em sua aplicação na abrangência de requisitos fundamentais para a modelagem desses sistemas.

O trabalho com as RdPC, mostra-se deveras promissor para a modelagem de OA, tirando proveito desta TDF ao aplicar significado às suas propriedades adequado à cada aplicação. Verificamos ao longo do trabalho a flexibilidade desta notação que traz em si uma robustez que atinge resultados além dos esperados de uma técnica de descrição formal tradicional.

Podemos, obedecendo esta mesma linha, gerar futuramente uma ligação mais estreita e formal entre as propriedades comportamentais e os objetivos cognitivos de um OA, ou ainda simplesmente às características fundamentais das interfaces digitais nas mais diversas áreas de aplicação.

8. Referencias Bibliográficas

Amaral, L. L., T. de A. Gomes, M. de F. C. de Souza, J. A. de Castro Filho, M. C. Pequeno. (2006). Um Aprimoramento do Modelo de Processo de Criação de Objetos de Aprendizagem do Projeto RIVED. WIE 2006 - Workshop de Informática na Educação. Campo Grande, MS, Brasil , 14-20 Julho, 2006.

Bell, T. E. e Thayer, T. A. (1976) Software Requirements: Are They Really a Problem? Proceedings on 2nd International Conference on Software Engineering. San Francisco.

Booch, G. J. Rumbaugh, Ivar Jacobson. (2005). The Unified Modeling Language User Guide (2nd Edition). Addison-Wesley Professional.

Castro J.F.B., (1995) Introdução a Engenharia de Requisitos, Mini-Curso JAI/SBC, RS, Agosto.

Galvis, A H. (1992). "Engenharia de Software Educativo". 1ª Ed. Santafé de Bogotá, Colômbia: Ediciones Uniandes – Universidade de Los Andes.

Gomes, A. S. e Wanderley E. G. (2003). Elicitando requisitos em projetos de software.

IEEE. Learning Technology Standardization Committee (LTSC). (2007) Acessado em: http://wiki.unifacs.br/gnufacs/twiki/bin/view/ObjetosDeAprendizagem/WebHom: 19 Mar.

Jensen.K, (1997) Coloured Petri Nets. Basic Concepts,Analysis Methods and Practical Use. Volume 1. EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin.

Monteiro, B.S., H. P. Cruz, M. Andrade, T. Gouveia, R. Tavares, L. F. C. Anjos. (2006). Metodologia de desenvolvimento de objetos de aprendizagem com foco na aprendizagem significativa. WIE 2006 - Workshop de Informática na Educação. Campo Grande, MS, Brasil , 14-20 Julho, 2006.

Prata, C. L., A. C. A. de A. Nascimento. (2007). Objetos de aprendizagem: uma proposta de recurso pedagógico/Organização: Carmem Lúcia Prata, Anna Christina Aun de Azevedo Nascimento. Brasília: MEC, SEED.

Souza, M. de F. C. de, D. G. Gomes, G. C. Barroso, C. T. de Souza, J. A. de Castro Filho, M. C. Pequeno, R. M. C. Andrade. (2007). LOCPN: Redes de Petri Coloridas na Produção de Objetos de Aprendizagem. Revista Brasileira de Informática na Educação, 2007.

Sommerville, I. Engenharia de Software - 8ª Edição (2007). Pearson Education.

Kirner, T.G., Davis, A.M, (1996) *"Nonfunctional Requirements of Real-Time Systems*", Advances in Computers (Zelkowitz, M.,Ed.), Volume 42, Academic Press, 1996, pp.1-37.

Wiley. D. A. (2007). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy in D. A. Wiley (Ed.), The Instructional Use of Learning Objects. http://reusability.org/read/chapters/wiley.doc, 15 Fev 2007.