

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO



**ÁRVORE DE CARACTERÍSTICAS E REDES DE PETRI
COLORIDA COM EXPRESSÕES DE LÓGICA
PROPOSICIONAL: PROPOSTAS DE MODELAGEM DE
REQUISITOS E FLUXO DE NAVEGAÇÃO**

CINTIA CARVALHO OLIVEIRA

Uberlândia - Minas Gerais

2011

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO



CINTIA CARVALHO OLIVEIRA

**ÁRVORE DE CARACTERÍSTICAS E REDES DE PETRI
COLORIDA COM EXPRESSÕES DE LÓGICA
PROPOSICIONAL: PROPOSTAS DE MODELAGEM DE
REQUISITOS E FLUXO DE NAVEGAÇÃO**

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Banco de Dados.

Orientador:

Prof. Dr. João Nunes de Souza

Co-orientador:

Prof. Dr. Renan Gonçalves Cattelan

Uberlândia, Minas Gerais

2011

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada “**Árvore de Características e Redes de Petri Colorida com Expressões de Lógica Proposicional: Propostas de Modelagem de Requisitos e Fluxo de Navegação**” por **Cintia Carvalho Oliveira** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 3 de Agosto de 2011

Orientador:

Prof. Dr. João Nunes de Souza
Universidade Federal de Uberlândia

Co-orientador:

Prof. Dr. Renan Gonçalves Cattelan
Universidade Federal de Uberlândia

Banca Examinadora:

Prof. Dr. Michel dos Santos Soares
Universidade Federal de Uberlândia

Prof. Dr. Renato Bulcão Neto
Universidade Federal de Goiás

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO

Data: Agosto de 2011

Autor: **Cintia Carvalho Oliveira**
Título: **Árvore de Características e Redes de Petri Colorida com Expressões de Lógica Proposicional: Propostas de Modelagem de Requisitos e Fluxo de Navegação**
Faculdade: **Faculdade de Computação**
Grau: **Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

Autor

O AUTOR RESERVA PARA SI QUALQUER OUTRO DIREITO DE PUBLICAÇÃO DESTE DOCUMENTO, NÃO PODENDO O MESMO SER IMPRESSO OU REPRODUZIDO, SEJA NA TOTALIDADE OU EM PARTES, SEM A PERMISSÃO ESCRITA DO AUTOR.

Resumo

Geralmente, a análise de requisitos é feita por proposições *ad hoc* de especialistas. O personagem que detém o conhecimento a respeito do domínio a ser desenvolvido, normalmente, não compreende a linguagem dos especialistas. Assim, as pessoas que deveriam especificar o software são relegadas a um segundo plano, como é o caso de alunos e professores do ensino fundamental em relação aos softwares educativos. Nesta dissertação, apresenta-se uma proposta para modelagem de requisitos, com o diferencial de se tratar de uma metodologia que pode ser usada por usuários. É utilizada uma concepção baseada na metodologia de desenvolvimento guiada por características, chamada de árvore de características.

Para desenvolver um sistema com qualidade é necessário uma eliciação de requisitos que consiga prever o máximo de características e um bom projeto e diagramas UML que reflitam o funcionamento do sistema. Porém cada vez mais a área de “Interação Humano-Computador” tem ganhado importância, pois um bom projeto de interface gráfica melhora a qualidade e usabilidade de um sistema, que envolve a aprendizagem, memorização e facilidade no uso. Para que um sistema possa garantir qualidade em sua interface são necessários planejamento e teste antes que essa interface seja efetivamente desenvolvida. A UML é uma linguagem de modelagem utilizada para planejar todo o funcionamento do sistema, porém ela não foi planejada para modelagem do fluxo de navegação de sistemas. Existem diversos trabalhos que se propõem a desenvolver métodos de modelagem de fluxo de navegação em interfaces, evidenciando uma necessidade crescente no planejamento e teste de interfaces, pois muitos sistemas falham devido a problemas na interação do usuário com o sistema.

O modelo de fluxo de navegação propõe uma análise de diferentes aspectos que fazem parte da modelagem, simulação e validação da interface de sistemas computacionais centrados no usuário. Nesta dissertação é proposto um modelo formal, baseado em Rede de Petri Colorida que auxilia o planejamento de interfaces, com foco principal no fluxo da navegação.

Palavras chave: eliciação de requisitos, modelagem de interface, redes de petri, lógica proposicional, interação humano-computador

Abstract

Requirement analysis usually consists of experts' ad hoc prepositions, however, the key character that holds the knowledge about the domain to be developed, usually does not understand the language of specialists. Thus, people who should specify the software are relegated to the background, as is the case of students and teachers of elementary schools regarding educational software. This paper presents a proposal for requirements elicitation with the peculiarity that it comprises a methodology that can be used by the user or even an important tool for developer and clients. Our proposal is based on the development methodology driven by features which we call tree features. To develop a

quality system is requirements elicitation that can predict the maximum features and a good design and UML diagrams that reflect the system operation. However increasingly the area of "Human-Computer Interaction" has gained importance as a well-designed graphical user interface, improves the quality and usability of a system, which involves learning, memory and ease of use. For a system to ensure quality in its interface design and testing are needed before this interface is actually developed. The UML is a modelling language used to plan the entire system operation, but it is not designed to model the system navigation flow. There are many works that propose to develop methods for modelling the interfaces navigation flow, indicating a growing need in the planning and testing interfaces, as many systems fail due to problems on user interaction with the system. The navigation flow model proposes an analysis of different aspects that are part

of the modelling, simulation and validation of user-centered computer systems interface. In this paper we propose a formal model based on Coloured Petri Net that helps the planning of interfaces, focusing primarily on the flow of navigation.

Keywords: requirements elicitation, modeling interface, petri net, propositional logic, human-computer interaction

Sumário

Lista de Figuras	xiii
Lista de Tabelas	xv
1 Introdução	17
1.1 Contexto	17
1.2 Motivação e Problema	18
1.3 Objetivos e Contribuições	18
1.4 Organização do Trabalho	19
2 Fundamentos	21
2.1 Engenharia de Requisitos	21
2.2 Redes de Petri	23
2.2.1 Elementos Básicos de Rede de Petri	23
2.2.2 Representação Matricial	27
2.2.3 Propriedades de Rede de Petri	29
2.2.4 Métodos de análise das propriedades	31
2.3 Rede de Petri Colorida (RdPC)	31
2.4 Interação Humano-Computador - IHC	33
2.4.1 Métodos de Modelagem de Interação	36
2.4.2 Métodos de Inspeção de Usabilidade	38
2.5 Lógica Proposicional	41
2.5.1 Síntese dos Conectivos da Lógica Proposicional	42
2.5.2 Semântica dos conectivos da Lógica Proposicional	43
3 Uma Proposta para Elicitação de Requisitos pelo Usuário: Árvore de Características	45
3.1 Árvore de Características	46
3.2 Modelando uma comunidade virtual de aprendizagem	47
3.3 Experimentação e Resultados	50
3.3.1 Perfil dos Participantes	50
3.3.2 Análise dos Resultados	51

3.3.3	Comentários dos participantes	52
3.3.4	Incorporação dos resultados	53
4	Rede de Petri Colorida com expressões da Lógica Proposicional: Modelagem de Fluxo de Navegação em Interfaces de Software	55
4.1	Levantamento de Necessidades em um Processo de Design de Interface Gráfica	56
4.2	Grafo de Telas	57
4.3	Rede de Petri Colorida com expressões da Lógica Proposicional	59
4.3.1	Linguagem da Lógica Proposicional utilizada na RdPCeLP	60
4.3.2	Representação Gráfica da Rede de Petri Colorida com expressões da Lógica Proposicional	63
4.3.3	Semântica da Linguagem da Lógica Proposicional utilizada na RdPCeLP	63
4.3.4	Condição de disparo da transição	67
4.3.5	Resultado do disparo de uma transição	68
4.4	Exemplo de Rede de Petri Colorida com Expressão da Lógica Proposicional	72
4.5	Definição Formal de RdPCeLP	74
4.6	Estudo de Caso representado pela Rede de Petri Colorida com expressões da Lógica Proposicional	75
4.7	Propriedades da RdPCeLP	77
4.8	Análise das propriedades da RdPCeLP	78
4.9	Simulação em Rede de Petri das regras de ouro e heurísticas	79
5	Trabalhos Relacionados	85
5.1	LOCPN (<i>Learning Objects production with Colored Petri Nets</i>)	86
5.2	PDIWA (<i>Processo de Design de Interface Web Adaptativa</i>)	87
5.3	MoLIC (<i>Modeling Language for Interaction as Conversation</i>)	89
5.4	OOHDM (<i>Object-Oriented Hypermedia Design Method</i>)	90
6	Conclusão	93
6.1	Síntese do Trabalho e Discussão	93
6.2	Trabalhos Futuros	94
	Referências Bibliográficas	97

Lista de Figuras

2.1	Rede de Petri Ordinária	24
2.2	Operações com mais recursos	25
2.3	Rede de Petri	27
2.4	Exemplo de <i>deadlock</i>	30
2.5	Rede de Petri Colorida	33
2.6	Processo de design de interfaces	35
3.1	Parte do modelo específico inicial para ensino de lógica à distância	47
3.2	Modelo do “Portal de Ensino”	48
3.3	Modelos das características	49
3.4	Exemplo de uma árvore de características de uma loja virtual	51
3.5	Ambiente Educacional por uma professora de Inglês	52
3.6	Árvore genérica por uma professora de Educação Física	52
3.7	Níveis de dificuldade encontrados para realizar a modelagem	53
3.8	Árvore de características resultante	54
4.1	“Corte” das características que não constituem telas do sistema	58
4.2	Grafo de Telas	59
4.3	Representação gráfica dos nós da RdPCeLP	64
4.4	Lugar Lg1	64
4.5	Lugar Lg2	65
4.6	Arco de entrada rotulado pela cláusula Ce	66
4.7	Arco de saída rotulado pela cláusula Cs	66
4.8	Esquema geral da Rede de Petri Colorida com Expressão Lógica Proposicional	67
4.9	Arco de chegada em uma transição	68
4.10	Disparo de uma transição	70
4.11	Exemplo de RdPCeLP	71
4.12	Arco rotulado com uma cláusula	72
4.13	Exemplo de Rede de Petri Colorida com expressões Lógicas	72
4.14	Modelo após o disparo de tr_1	74
4.15	RdPCeLP da base da Comunidade Virtual de Aprendizagem	76

4.16 RdPCeLP de Chat	77
4.17 Simulação da rede, cenário antes do disparo de tr_1	79
4.18 Simulação da rede, cenário depois do disparo de tr_1	79
4.19 Atalho para recursos de Fórum	80
4.20 Rede de Petri para envio de formulário de contato	81
4.21 Rede de Petri para confirmação de envio de formulário de contato	82
5.1 Representação de uma Transição entre telas	86
5.2 Representação Hierárquica da Estrutura de um Objeto de Aprendizagem	87
5.3 Modelo do Processo de Design de Interface web Adaptativa	88
5.4 Um diagrama de diálogo básico em MoLIC	90
5.5 Exemplo de Modelagem OOHDM	91
6.1 Exemplo de geração de erro	95

Lista de Tabelas

2.1	Simbologia para a Modelagem de Características	22
2.2	Interpretação a respeito de transições e lugares	26
2.3	Tabela-verdade da fórmula $P \wedge Q$	44
2.4	Tabela-verdade da fórmula $P \vee Q$	44
2.5	Tabela-verdade da fórmula $P \rightarrow Q$	44
2.6	Tabela-verdade da fórmula $P \leftrightarrow Q$	44

Capítulo 1

Introdução

Este trabalho propõe uma análise de diferentes aspectos que fazem parte da modelagem, simulação e validação da interface de sistemas computacionais centrados no usuário. Neste trabalho é analisado um modelo formal que auxilia o planejamento de interfaces de sistemas computacionais, com foco principal no fluxo da navegação.

1.1 Contexto

Projetar software é construir modelos que representem os vários aspectos de um sistema computacional muitas vezes dotado de uma interface com o usuário. Na engenharia de software são estudados e desenvolvidos modelos que representam as características e funcionalidades de um sistema computacional.

Com o intuito de representar a estrutura e comportamento de um software, a UML (*Unified Modeling Language*), por exemplo, foi adotada como linguagem padrão na elaboração de modelos [Booch et al. 2000]. De acordo com [Paternò 2001], a UML é a abordagem baseada em modelos mais bem sucedida para apoiar o desenvolvimento de software. Porém, de acordo com [Silva e Silveira 2010], durante a sua evolução houve pouca preocupação no projeto e desenvolvimento de interfaces com o usuário.

Segundo [Machado 2006] a UML padrão não abrange modelos cujo foco é a modelagem, simulação e validação do fluxo de navegação em interfaces gráficas de sistemas de informação. A UML tem sido amplamente aceita pelos desenvolvedores de aplicações, porém não tanto pelos desenvolvedores de interface [Silva e Paton 2000].

De acordo com [Silva e Paton 2000] a interface de um software representa uma quantidade significativa de código implementado, porém não existe uma linguagem de modelagem ou ferramenta que suporta a colaboração entre os desenvolvedores de interface e os desenvolvedores da aplicação.

Em geral, quando no projeto de interfaces de software com o usuário, os projetistas utilizam técnicas estudadas na área de “Interação Humano-Computador”, mas não há um

modelo padrão a ser seguido. Geralmente os projetistas se baseiam em protótipos não funcionais para simular o comportamento visual do software.

Existem instrumentos oficiais (normas ISO) e não oficiais que estabelecem normas, princípios, diretrizes, critérios, recomendações, regras, métodos, que são úteis e apoiam os projetistas, por contemplarem desde a concepção até a avaliação das interfaces com o usuário [Batista e Ulbricht 2010].

1.2 Motivação e Problema

Um bom projeto de interface gráfica de um sistema computacional, que tenha interação com o usuário, melhora a qualidade no uso de um sistema e a usabilidade que envolve a aprendizagem, memorização e facilidade no uso. Isso porque sistemas que interagem com o usuário têm na interface sua base de comunicação. E, nesse caso, a navegabilidade deve ser provida de forma cuidadosa, pois muitos erros partem, por exemplo, de telas inalcançáveis. São também erros em comandos, operações que não podem ser efetuadas, entre outros.

Em [Campos 2004], foi realizada uma pesquisa na qual se estima que 60% a 90% dos sistemas falham por conta de problemas na interação do usuário com a interface do sistema.

Modelos gráficos e matemáticos baseados, por exemplo, em Redes de Petri [Murata 1989] podem auxiliar no planejamento de interfaces de sistemas computacionais. Tais modelos permitem uma melhor análise e compreensão do funcionamento externo de um sistema, e também a validação dos requisitos na interface. Eles permitem gerar projetos computacionais com mais qualidade, diminuindo a chance de erro e incrementando a usabilidade dos sistemas computacionais centrados no usuário. Isso porque as características das Redes de Petri permitem uma modelagem navegacional e da interação, o que facilita a verificação e simulação dos sistemas modelados.

Uma Rede de Petri (RdP) é, de acordo com [Murata 1989], uma ferramenta promissora para descrever e estudar sistemas de processamento de informação que sejam concorrentes, assíncronos, distribuídos, paralelos e estocásticos - características observadas também na modelagem do fluxo de navegação em interfaces gráficas.

1.3 Objetivos e Contribuições

Uma questão fundamental no desenvolvimento de sistemas é o levantamento dos requisitos do mesmo. Neste trabalho são descritos dois tipos de modelagem: um modelo de requisitos de sistema e um modelo para fluxo de navegação de software. A partir do modelo de requisitos, são extraídas as informações necessárias para a elaboração do modelo

de fluxo navegacional.

Além do foco na modelagem no fluxo de navegação em sistemas centrados no usuário, este trabalho objetiva oferecer ao projetista e ao usuário, ou cliente do sistema, uma modelagem de requisitos denominada *árvore de características*. Os resultados da aplicação desse modelo com o usuário foram publicados em [Oliveira et al. 2010d].

Nesse contexto, a modelagem baseada em RdP pode ser utilizada para a resolução de problemas que se fundamentam no fluxo de informação e encadeamento de ações, o que configura, basicamente, a solução dos problemas próprios da modelagem de interfaces de sistemas computacionais.

Dessa forma, o principal objetivo deste trabalho é, portanto, considerar as propriedades das RdP para modelar, descrever e estudar sistemas de processamento de informação no que se refere ao fluxo das interfaces gráficas com o usuário.

Tais interfaces devem possuir um bom planejamento, fluidez no seu uso e minimizar os erros do usuário. E para que esses objetivos sejam alcançados, é proposto ao projetista um método formal que facilita o planejamento, a geração, a simulação e a validação dos modelos obtidos. Esse modelo é baseado em RdP Coloridas, uma extensão das RdP com expressões da Lógica Proposicional. O intuito é oferecer semântica e testes condicionais para a organização do fluxo de navegação de interface.

Dessa forma, foram unidas em um único modelo as RdP Coloridas e a Lógica Proposicional, criando-se o conceito de Rede de Petri Colorida com expressões da Lógica Proposicional, ou RdPCeLP.

Por fim, dados os modelos propostos, é feita uma análise de suas virtudes e limitações. E, a partir dela, são também analisadas algumas possíveis modificações dos modelos formais propostos.

1.4 Organização do Trabalho

O restante desta dissertação encontra-se organizado da seguinte maneira: no Capítulo 2 são apresentados os fundamentos de Engenharia de Requisitos, Rede de Petri, Rede de Petri Colorida, Interação Humano-Computador e Lógica Proposicional; no Capítulo 3 é apresentada a abordagem desenvolvida para eliciação de requisitos pelo usuário, denominada de Árvore de Características; no Capítulo 4 é apresentada a técnica de modelagem de fluxo de navegação com a RdP desenvolvida, denominada de Rede de Petri Colorida com expressões da Lógica Proposicional; no Capítulo 5 são apresentados os trabalhos relacionados; e, por fim, no Capítulo 6 são apresentadas as conclusões e sugeridos trabalhos futuros.

Capítulo 2

Fundamentos

Nesse capítulo são introduzidos os conceitos básicos utilizados para a concepção do trabalho ora apresentado.

2.1 Engenharia de Requisitos

Segundo [Sommerville 2007], o processo de descobrir, analisar, documentar e verificar serviços e restrições é chamado de *engenharia de requisitos*. Ele define ainda que *requisitos* de um sistema são descrições dos serviços fornecidos pelo sistema e suas restrições operacionais.

De acordo com [Silva 2005] existem muitas técnicas de modelagem de requisitos, sendo as seguintes as mais expressivas:

- **Casos de Uso** Casos de Uso são conjuntos de sequências de ações, resultantes da interação do sistema com um ator [Booch et al. 2000];
- **Viewpoints** Viewpoints consideram aspectos do sistema percebidos por diferentes requerentes [Finkelstein et al. 1992];
- **Expressão Controlada de Requisitos - CORE** As perspectivas podem ser físicas ou funcionais, representando organizações, homens, entidades de hardware ou software, e processos; no final combinadas representam as atividades [Finkelstein et al. 1992];
- **Diagrama de Fluxo de Dados** O Diagrama de Fluxo de Dados é uma representação gráfica que mostra o fluxo de informação e as transformações que são aplicadas à medida que os dados se movem da entrada para a saída [Pressman 2006];
- **Processador de Linguagem de Requisitos - RLP** Desenvolvimento de lingua-

gens formais para os requisitos de especificação das aplicações [Silva et al. 2005];

- **Metodologia da Engenharia de Requisitos de Software - SREM** Os caminhos dos dados consistem nas mensagens de entrada, na sequência das tarefas processadas que envolvem o fluxo do controle, e nas mensagens de saída [Finkelstein et al. 1992];
- **Volere** Volere é um método completo de obtenção de requisitos, baseado nos casos de uso [Fisher 2001].

Neste trabalho é proposta uma estratégia para definição dos requisitos, que se fundamenta na metodologia de desenvolvimento guiado por características [Filho et al. 2004] (*Feature Driven Development* - FDD), a qual é referenciada por árvore de características.

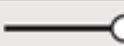
No desenvolvimento guiado por características, tem-se a capacidade de especificar a variabilidade de propriedades (os requisitos) e suas interdependências através de um modelo.

Nesse contexto, uma característica é uma função valorizada pelo cliente que deve ser construída e acoplada ao novo sistema. Essa característica é expressa pelo cliente em uma linguagem natural, o que “facilita” a comunicação entre os envolvidos no projeto de construção do novo sistema.

Conforme [Filho et al. 2004], a utilização desse paradigma de desenvolvimento propicia a geração de software por meio da composição de características. Outra vantagem dessa abordagem, além da correlação entre características, é sua representação gráfica. Em geral, para o usuário final, a representação gráfica é importante por ser um modelo semântico que descreve as relações estruturais entre as características de um domínio de aplicação.

Uma simbologia para a representação do modelo, utilizada por [Filho et al. 2004], é mostrada na Tabela 2.1, na qual se observa que a notação utiliza retângulos para representar as características e quatro tipos de arcos que interligam as características pai e filhas indicando que as características “filhas” podem ser obrigatórias, alternativas, opcionais ou dependentes.

Tabela 2.1: Simbologia para a Modelagem de Características.

 Nome	Característica	
	Obrigatórias	Deve estar presente
	Alternativas	Deve estar presente em um e somente um
	Opcionais	Pode ou não estar presente
	Dependentes	Depende de uma outra características existente.

As características *obrigatórias* devem estar presentes no sistema. As características *alternativas* devem aparecer somente uma de cada vez, ou seja, é um OU exclusivo entre as características apontadas por este arco. As *opcionais* podem ou não estar presentes, enquanto que as características *dependentes* não se relacionam necessariamente entre pai e filhas, e indicam que, para a característica existir, a que está interligada a ela pelo arco também deve existir no sistema.

2.2 Redes de Petri

Redes de Petri (RdP) são ferramentas de modelagem gráfica e matemática. Como ferramenta gráfica elas podem ser utilizadas para comunicação visual, similar aos diagramas de fluxo de dados e redes. Além disso, elas podem simular as atividades dinâmicas e concorrentes de sistemas. Como ferramenta matemática, é possível definir um conjunto de equações algébricas e outros modelos matemáticos [Murata 1989].

O conceito de RdP teve origem na dissertação de Carl Petri [Petri 1966], submetida em 1962, intitulada *Kommunikation mit Automaten* (Comunicação com Autômatos), defendida na Faculdade de Matemática e Física da Universidade Técnica de Darmstadt, na Alemanha.

2.2.1 Elementos Básicos de Rede de Petri

Os elementos básicos de uma Rede de Petri são os lugares, transições, fichas e arcos:

- **Lugares** Os lugares são representados por um círculo e podem ser considerados como estados dentro do sistema.
- **Transições** As transições são representadas por uma barra vertical ou um retângulo e podem ser consideradas como eventos que ocorrem no sistema.
- **Fichas** As fichas são representadas por pontos em um lugar e podem ser consideradas como se a condição associada ao lugar é verificada.
- **Arcos** Arcos são representados por linhas que ligam lugares em transições e transições em lugares. E representam o fluxo de comunicação ou controle dentro do sistema.

Os lugares e fichas podem receber muitas interpretações, podendo ser entendidos como entidades físicas ou abstratas.

De acordo com [Cardoso e Valette 1998], com a RdP pode-se chegar a uma abstração completa dos aspectos descritivos do sistema (fichas e lugares). Essa abstração permite uma visão sintética do sistema a ser modelado.

O funcionamento de uma RdP é baseado no disparo de transições. Esse *disparo* consiste em um evento que ocorre quando os lugares que estão ligados em transições (lugares de entrada) possuem fichas. O disparo de uma transição consiste na execução de dois passos [Cardoso e Valette 1998]:

- De cada lugar de entrada retira-se uma ficha, indicando assim que a condição anterior à transição não é mais verdadeira, caso esse lugar fique sem nenhuma ficha;
- Depositam-se as fichas nos lugares de saída, indicando que, após a ocorrência do disparo da transição, as atividades indicadas por esses lugares serão executadas e o próximo estado será verdadeiro.

Na Figura 2.1(a), a transição simula o início da operação de uma máquina em uma indústria. O evento *iniciar operação* está associado à transição tr_1 . O disparo dessa transição somente ocorre se houver pelo menos uma ficha nos lugares *máquina disponível*, *operador disponível* e *peças*. As fichas nesses lugares indicam que as condições dos lugares são verdadeiras, ou seja, o modelo representa que na operação dessa Rede de Petri existe uma máquina disponível, um operador disponível e uma peça a ser processada.

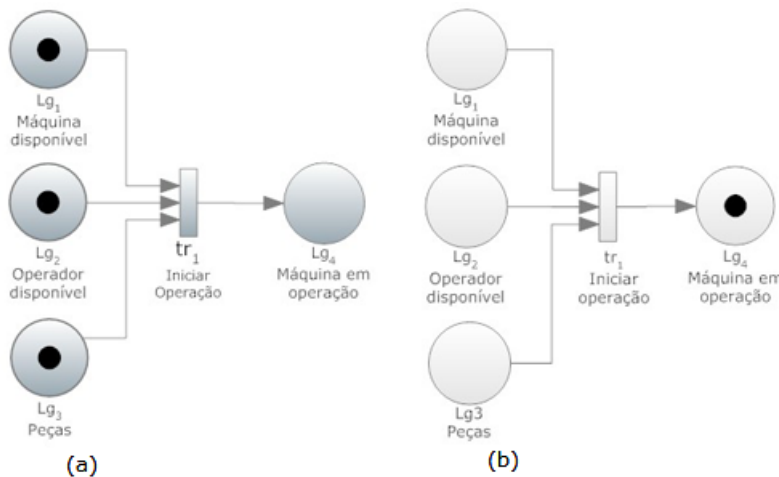


Figura 2.1: Rede de Petri Ordinária

Com os lugares de entrada satisfazendo a condição de disparo da transição (pelo menos uma ficha em cada lugar), o evento *iniciar operação* ocorrerá, consistindo na retirada de uma ficha de cada lugar de entrada e a colocação de uma ficha nos lugares de destino, ou seja, no lugar *máquina em operação*, como descrito na Figura 2.1(b).

A semântica do modelo apresentado propõe representar a seguinte situação: para que uma peça seja processada, deve existir uma peça, e é necessário que uma máquina e um operador estejam disponíveis. Tendo esses pré-requisitos satisfeitos, o disparo da transição simboliza que um evento ocorrerá, ou seja, que a próxima condição após o disparo será verdadeira, o que é caracterizado pela *máquina em operação*.

A retirada das fichas dos lugares de entrada, após o disparo da transição, indica que os estados ou condições representados por eles não são mais verdadeiros.

Entretanto, se antes do disparo da transição há mais de uma ficha no lugar de entrada, então a condição representada pela ficha ainda continua verdadeira. A condição somente se torna falsa no caso em que o lugar de entrada fica vazio após o disparo da transição.

Nas Redes de Petri ordinárias, as fichas funcionam como ativadores de estados e disparo de transições, coordenando as operações representadas pela Rede de Petri. Observa-se que os arcos que ligam lugares em transições e transições em lugares podem possuir um peso. Esse peso indica quantas fichas devem passar pelo arco, tanto para corresponder a uma condição, quanto na geração de fichas.

Nas Redes de Petri de alto nível (redes coloridas e predicado-transição, por exemplo) as fichas podem representar informação, e não servem somente para controle (como é o caso das Redes de Petri ordinárias).

Na Figura 2.2 é representado um problema de fluxo de um processo, no qual temos 2 peças que pertencem a 2 processos diferentes de fabricação. Em cada processo são executados 2 operações (**Proc1** e **Proc2**), por duas máquinas diferentes M_1 e M_2 . Em **Proc1**, a ordem de passagem pelas máquinas é M_1 depois M_2 . Já em **Proc2**, a ordem de passagem é M_2 depois M_1 .

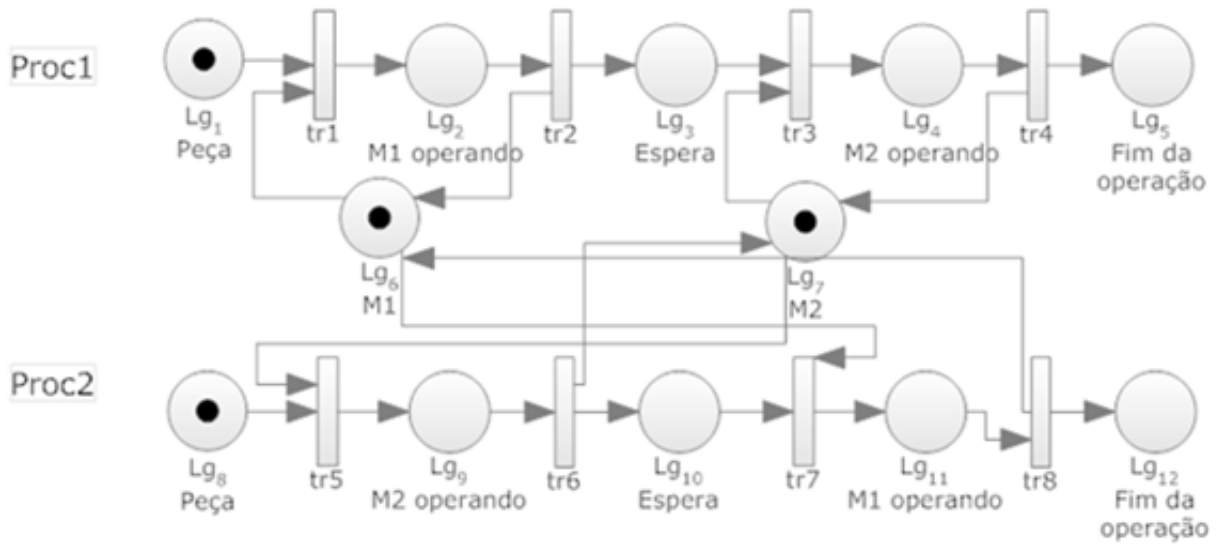


Figura 2.2: Operações com mais recursos

O funcionamento da Rede de Petri da Figura 2.2 é descrito a seguir:

Os lugares Lg_1 , Lg_8 , Lg_6 e Lg_7 possuem fichas. Isso significa que as condições definidas por eles são verdadeiras. Como os lugares Lg_1 e Lg_8 são rotulados por **Peça**, isso significa que eles são responsáveis por demonstrar a presença, ou não, de alguma peça a ser processada.

Se existe uma ficha no lugar, então a peça deve ser processada. Caso contrário, ela não deve ser processada. Já os lugares Lg_6 e Lg_7 , rotulados por M_1 e M_2 , respectivamente,

Tabela 2.2: Interpretação a respeito de transições e lugares.

Lugares de entrada na transição	Transições	Lugares de Saída da Transição
Dados de entrada	Processamento Computacional	Dados de saída
Sinais de entrada	Processamento de sinais	Sinais de saída
Necessidades de Recursos	Tarefa ou trabalho	Liberção de Recursos
Condições	Cláusulas lógicas	Conclusões
Buffers	Processador	Buffers

demonstram que as duas máquinas estão disponíveis para uso.

Como os arcos não possuem nenhum rótulo indicando seu peso, isso corresponde a dizer que o peso do arco é igual a 1, por padrão. As transições tr_1 e tr_2 estão habilitadas para o disparo, pois existe ficha nos lugares que estão ligados aos arcos de entrada na transição. Com o disparo de tr_1 ocorre a geração de uma ficha que é colocada no lugar Lg_2 , indicando que a condição M_1 operando passou a ser verdadeira.

E, nesse caso, são retiradas uma ficha de cada lugar de entrada, Lg_1 e Lg_6 , indicando que o estado do sistema representado por elas passou a não existir, ou seja, não existe uma peça a ser processada e nem a máquina M_1 está disponível.

Além disso, é retirada somente uma ficha dos lugares Lg_1 e Lg_6 por que o peso do arco não está explícito. Isso significa que o peso é 1, indicando, assim, que somente uma ficha é necessária para satisfazer a condição de disparo em relação àquele lugar de entrada.

O funcionamento, ou fluxo das fichas, da Rede de Petri da Figura 2.2 prossegue até que as duas peças sejam processadas. Isso ocorre através de disparos sucessivos de transições, que geram e consomem fichas até que os lugares Lg_5 e Lg_{12} possuam, cada um, uma ficha.

Assim, o estado representado pelos rótulos de Lg_5 e Lg_{12} , ou seja, **Fim da operação**, indica que o processamento das peças foi finalizado.

O consumo e geração de fichas através do disparo das sucessivas transições representa a mudança de estados dentro de um sistema. Os estados ativos em um sistema são representados graficamente por um lugar com uma ficha, enquanto que os lugares que não possuem fichas são estados nos quais o sistema não se encontra ativo.

Na Tabela 2.2, adaptada de [Murata 1989], são apresentadas algumas interpretações que podem ser feitas a respeito das transições e lugares.

Definição 1 - Rede de Petri

A definição a seguir segue a notação e conceitos apresentados em [Murata 1989].

Uma Rede de Petri é uma quintupla, $PN = (P, T, F, W, M_0)$, na qual:

- $P = \{p_1, p_2, \dots, p_n\}$ é um conjunto finito de lugares
- $T = \{t_1, t_2, \dots, t_n\}$ é um conjunto finito de transições
- $F \subseteq \{P \times T\} \cup \{T \times P\}$ é um conjunto de arcos
- $W : F \rightarrow \{1, 2, 3, \dots\}$ é uma função peso

- $M_0 : P \rightarrow \{0, 1, 2, 3, \dots\}$ é a marcação inicial
- $P \cap T = \emptyset$ e $P \cap T \neq \emptyset$

2.2.2 Representação Matricial

Uma Rede de Petri pode ser representada utilizando uma notação matricial denominada de matriz de incidência, que representa a relação entre lugares e transições [Cardoso e Valette 1998, Pais 2004].

Definição 2 - Matriz de Incidência Prévia Dada uma Rede de Petri, a sua matriz de incidência prévia é denominada por **Pre**.

A matriz **Pre** tem dimensão $n \times m$, tal que n é o número de lugares e m é o número de transições. Cada elemento a_{ij} da matriz **Pre** corresponde ao peso do arco que interliga o lugar p_i à transição t_j . Portanto, um arco liga um lugar p_i a uma transição t_j se, e somente se, $\mathbf{Pre}[p_i, t_j] \neq 0$.

Exemplo 1

Considere a Rede de Petri representada na Figura 2.3.

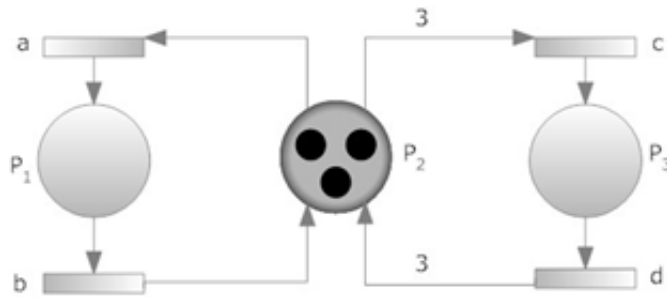


Figura 2.3: Rede de Petri, adaptada de [Cardoso e Valette 1998]

A matriz de incidência prévia é dada por:

$$Pre = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{matrix} P_1 \\ P_2 \\ P_3 \end{matrix} \end{matrix}$$

Os elementos da matriz com o valor 0 determinam que o lugar corresponde ao número da linha não está ligado na transição que corresponde ao número da coluna. Em outras palavras, tal lugar não é um lugar de incidência da transição.

Os demais elementos da matriz, que estão com algum valor diferente de zero, indicam que há a ligação do lugar que corresponde à linha com a transição indicada pela coluna. Nesse caso, o valor a_{ij} corresponde ao peso associado ao arco que liga o lugar p_i à transição t_j .

Definição 3 - Matriz de Incidência Posterior

Dada uma Rede de Petri, a sua matriz de incidência posterior é denominada por **Post**.

A matriz **Post** tem dimensão $n \times m$, tal que n é o número de lugares e m é o número de transições. Cada elemento b_{ij} da matriz **Post** corresponde ao peso do arco que interliga a transição t_j ao lugar p_i . Portanto, um arco liga uma transição t_j a um lugar p_i se, e somente se, $\mathbf{Post}[p_i, t_j] \neq 0$.

Exemplo 2

Considere a Rede de Petri representada na Figura 2.3.

A matriz de incidência posterior é dada por:

$$\mathbf{Post} = \begin{array}{cccc} & a & b & c & d \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \end{bmatrix} & P_1 & P_2 & P_3 \end{array}$$

Os elementos da matriz com o valor 0, apontam que a transição indicada pela coluna não está ligada no lugar indicado pela linha.

Os demais elementos da matriz que possuem algum valor indicam que há ligação da transição que é representada pela coluna com o lugar indicado pela linha.

Definição 4 - Matriz de incidência

A partir das matrizes **Pre** e **Post** pode-se definir a matriz de incidência $\mathbf{M}_{\text{inc}}$, que é resultado da subtração de **Post** por **Pre**. Assim:

$$\mathbf{M}_{\text{inc}} = \mathbf{Post} - \mathbf{Pre}$$

Na matriz de incidência, cada coluna corresponde à modificação da marcação através do disparo da transição associada.

Exemplo 3

Considerando a Rede de Petri representada na Figura 2.3, temos que $\mathbf{M}_{\text{inc}}$ é dado por:

$$\mathbf{M}_{\text{inc}} = \mathbf{Post} - \mathbf{Pre} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & -3 & 3 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

Então, por exemplo, a primeira coluna da matriz $\mathbf{M}_{\text{inc}}$, significa que quando a transição a é disparada é adicionada uma marca no lugar P_1 e removida uma marca do lugar P_2 .

Quando a transição c é disparada, é removida 3 fichas do lugar P_2 e adicionada uma ficha no lugar P_3 .

2.2.3 Propriedades de Rede de Petri

Conforme [Murata 1989], dois tipos de propriedades podem ser estudadas com modelos de Rede de Petri:

- propriedades comportamentais e
- propriedades estruturais.

O primeiro tipo depende da marcação inicial (disposição inicial das fichas), e o outro tipo é independente. As propriedades comportamentais são:

- **Alcançabilidade** (*reachability*)

Um lugar é dito alcançável caso exista uma sequência de disparos de transições a partir de uma marcação inicial M_0 que atinja esse lugar. Quando um lugar é alcançado, estabelece uma configuração de distribuição de fichas denominada por marcação M_n . O problema da alcançabilidade na Rede de Petri é encontrar M_n , tal que depois de uma sequência de disparos, M_0 se transforma em M_n . Uma sequência de disparos é denotada por:

$$\sigma = M_0 \ t_1 \ M_1 \ t_2 \ M_2 \ \dots \ t_n \ M_n$$

ou

$$\sigma = t_1 \ t_2 \ \dots \ t_n$$

A notação

$$[M_0, \sigma] \geq M_n \tag{2.1}$$

denota que M_n é alcançável de M_0 por σ .

Uma Rede de Petri é dita alcançável se a partir da marcação inicial e após uma sequência finita de disparos cada lugar da RdP pode ser alcançado.

- **Limitação** (*boundedness*)

Uma Rede de Petri é dita k -limitada se o número de fichas em cada lugar não ultrapassar um número finito k , ou seja, define o número de marcas que cada lugar pode acumular.

- **Vivacidade** (*liveness*)

Uma Rede de Petri é dita viva caso não ocorra, durante a operação da rede nenhum *deadlock*. Ou seja, em infinitos disparos de transição a Rede de Petri não para.

A Figura 2.4 ilustra um deadlock, sendo, portanto, uma rede que não possui a propriedade de vivacidade.

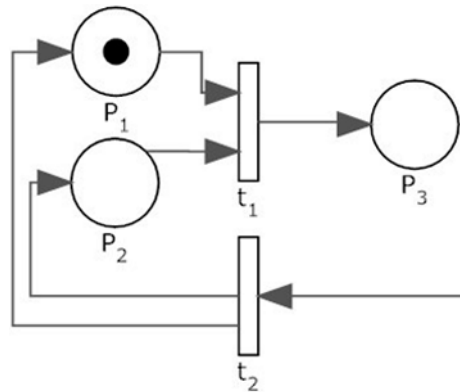


Figura 2.4: Exemplo de *deadlock*

A Figura 2.4 representa uma Rede de Petri em um dado momento de sua execução. Nesse caso, a transição t_1 não pode ser disparada, pois nem todos os estados de entrada da transição possuem uma ficha. Assim, é dito que uma Rede de Petri é viva quando não houver possibilidade de *deadlock*.

- **Reversibilidade (*reversibility*)**

Uma Rede de Petri é dita reversível caso a marcação inicial volte a ocorrer ao longo das sequências de disparo.

- **Cobertura (*coverability*)**

A cobertura identifica se uma transição é potencialmente disparável, isto é, se uma marca pode ser obtida a partir de outra anterior a ela [Rieder 2007].

- **Persistência (*persistence*)**

Uma Rede de Petri é dita persistente se, para duas transições que podem ser disparadas, o disparo de uma delas não irá impedir o disparo da outra. Assim, uma transição em uma rede persistente, uma vez habilitada para o disparo, permanecerá assim até que dispare.

- **Distância sincrônica (*synchronic distance*)**

A Distância sincrônica é uma métrica relacionada com o grau de dependência mútua entre dois eventos.

- **Justiça (*fairness*)**

Uma Rede de Petri é dita justa se para quaisquer duas transições, o número de vezes que uma é executada enquanto a outra não é executada é finito [Rieder 2007].

2.2.4 Métodos de análise das propriedades

Como uma Rede de Petri pode ser formulada matematicamente, então suas propriedades podem ser verificadas.

Existem algumas formas de análise do comportamento de uma Rede de Petri [Murata 1989, Pais 2004], dentre elas:

- Análise por enumeração de estados;
- Técnicas de redução;
- Matriz de incidência e equações de estado;
- Simulação

Análise por enumeração de estados

Pela análise por enumeração de estados é construído um grafo de alcançabilidade. Esse grafo efetua a enumeração de todas as marcações possíveis ou alcançáveis a partir da marcação inicial M_0 .

No grafo de alcançabilidade cada nó representa uma marcação possível dentro da rede e cada arco traduz o disparo de uma transição ou transições que estejam em concorrência.

Com a construção desse grafo é possível verificar todas as propriedades comportamentais de uma Rede de Petri, porém é uma abordagem que muitas vezes é impraticável, por conta da geração de muitas marcações, o que acarreta em uma “explosão” de nós.

Técnicas de redução

As técnicas de redução consistem na redução de um modelo complexo a um mais simples e fácil de tratar, porém sem que haja perda de propriedade.

Essa técnica consiste na substituição de sub-redes, porém não pode ser aplicada a muitos tipos de Redes de Petri.

Matriz de incidência e equações de estado

A matriz de incidência e equações de estado permite estudar as propriedades comportamentais de uma Rede de Petri através de suas representações algébricas. Utiliza-se a matriz de incidência para descrever a ligação entre lugares e transições.

Simulação

A simulação é utilizada quando se tem uma Rede de Petri relativamente complexa, o que dificulta sua análise por métodos analíticos.

A simulação não garante que todas as propriedades testadas estão presentes, porém pode certificar que para os cenários estudados elas estão ou não presentes.

2.3 Rede de Petri Colorida (RdPC)

A proposta apresentada neste trabalho é uma aplicação de Redes de Petri Colorida, cuja definição é considerada a seguir.

Uma Rede de Petri Colorida é uma extensão da Rede de Petri, sendo caracterizada por diferenciar as fichas, para isso elas são associadas a cores, que definem um significado a cada ficha, seja esse significado uma função ou informação.

De acordo com [Kristensen et al. 1998] a Rede de Petri Colorida provê um *framework* para a construção e análise de sistemas distribuídos e concorrentes, provendo a união do poder de representação das Redes de Petri e o poder das linguagens de programação.

A Rede de Petri provê as estruturas para descrever a sincronia de processos concorrentes, enquanto a linguagem de programação provê os tipos de dados (conjunto de cores) e a manipulação de dados.

As Redes de Petri Coloridas permitem que valores de dados sejam atribuídos às fichas, e que elas sejam diferenciadas umas das outras através de cores.

De acordo com [Ramos e Oliveira 2009], comparando as Redes de Petri Coloridas com as clássicas, ou ordinárias, algumas características podem ser identificadas:

- Cada Rede de Petri Colorida pode ser transformada em uma Rede de Petri ordinária equivalente e vice-versa;
- Não existe ganho teórico ao utilizar Redes de Petri Coloridas, porém na prática elas são mais compactas e convenientes na modelagem que as redes clássicas;
- O formato de representação gráfica facilita a compreensão, o que permite uma melhor aceitação pelos analistas de sistemas, pois se assemelha aos diagramas de fluxo de dados;
- As representações gráficas que representam os lugares, estados, arcos e fluxo de informação das Redes de Petri clássicas também são adotados pelas RdPC.

Definição 5 - Rede de Petri Colorida

A definição a seguir segue a notação e conceitos apresentados em [Jensen 1992].

Uma Rede de Petri Colorida é uma 9-tupla:

$$RdPC = \{\Sigma, P, T, A, N, C, G, E, I\}$$

sendo:

- Σ : conjunto finito de cores ou tipos, não nulo;
- P : conjunto finito de lugares;
- T : conjunto finito de transições;
- A : conjunto finito de arcos;
- N : $A \rightarrow (P \times T) \cup (T \times P)$, representa a função de nó;
- C : $P \rightarrow \Sigma$, representa a função de coloração;

- G: função de controle;
- E: representa a função expressão de arco;
- I: é a função de inicialização definida em P.

O exemplo da Figura 2.5 considera o mesmo problema representado pela Rede de Petri da Figura ???. Nesse caso, o problema é representado por uma Rede de Petri Colorida, com três máquinas e três peças [Cardoso e Valette 1998].

- O conjunto de cores do lugar PC é o conjunto $\{pc1, pc2, pc3\}$;
- O conjunto de cores do lugar MAQ é o conjunto $\{maq1, maq2, maq3\}$;
- As cores do lugar *US* formam o conjunto $\{pc1.maq1, pc2.maq1, pc3.maq1, \dots, pc3.maq3\}$, que são relativas as diferentes operações de usinagem das peças.

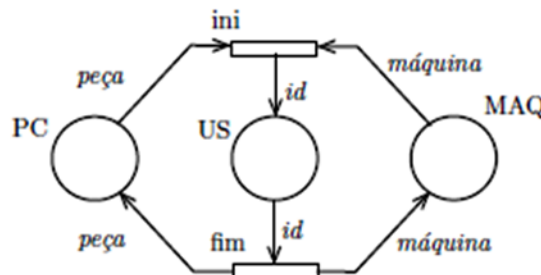


Figura 2.5: Rede de Petri Colorida [Cardoso e Valette 1998]

Como as fichas de uma Rede de Petri Colorida possuem semântica, então, em geral tais redes são menores, ou mais simples que as Redes de Petri ordinárias.

No exemplo considerado, a rede da Figura 2.5 é um grafo mais simples, uma vez que as fichas possuem uma semântica associada. Isso permite uma associação de fichas correspondentes à peça e máquina.

2.4 Interação Humano-Computador - IHC

A utilização de uma interface gráfica muitas vezes não atende as necessidades e expectativas do usuário final. Esse problema impacta diretamente no aumento do índice de retrabalho de uma equipe de desenvolvimento de um sistema, e pode ainda acarretar abandono do sistema pelo usuário [Cordeiro et al. 2009].

Para um sistema computacional que possui uma interface com o usuário, além de oferecer todas as funções para as quais foi projetado, é fundamental que tenha um *design* adequado para sua utilização de forma fácil e prática.

A área que estuda a interação do usuário com sistemas computacionais é a **Interação Humano-Computador**.

De acordo com [Rocha e Baranauskas 2003]:

Os objetivos de IHC são o de produzir sistemas usáveis, seguros e funcionais. Esses objetivos podem ser resumidos como desenvolver ou melhorar a segurança, utilidade, efetividade e usabilidade de sistemas que incluem computadores. Nesse contexto o termo sistemas se refere não somente ao hardware e o software, mas a todo o ambiente que usa ou é afetado pelo uso da tecnologia computacional.

No projeto de uma interface, o projetista deve levar em conta a qualidade da interação do usuário com o sistema, especialmente no que tange a facilidade e eficiência com a qual um usuário utiliza o sistema.

Para se avaliar a usabilidade de um sistema alguns fatores devem ser levados em conta [Nielsen 1993, Preece et al. 2002]:

- **Facilidade de aprendizagem**

A facilidade de aprendizagem se refere à necessidade de tempo e esforço dos usuários para aprender a utilizar um sistema;

- **Facilidade de uso**

A facilidade de uso se refere ao esforço cognitivo para interagir com o sistema, como também a facilidade de completar uma tarefa sem erros;

- **Eficiência de uso e produtividade**

A eficiência de uso analisa se o sistema consegue fazer bem aquilo a que se propõe e a produtividade se refere à execução das tarefas de forma rápida e eficaz;

- **Satisfação do usuário**

A satisfação do usuário é uma avaliação subjetiva do sistema, ela se refere às preferências pessoais e emoções dos usuários na interação com o sistema;

- **Flexibilidade**

A flexibilidade de um sistema é a capacidade de proporcionar caminhos distintos para se atingir um objetivo, se adaptando assim ao modo de trabalho individual dos usuários;

- **Utilidade**

A utilidade se refere ao conjunto de funcionalidades que são oferecidas por um sistema;

- **Segurança no uso** A segurança no uso garante proteção na utilização de um sistema, envolvendo desde recuperação de erros até o impacto do uso do sistema na saúde dos usuários.

De acordo com [Nielsen 1993, Preece et al. 2002] a usabilidade foi proposta com base no conhecimento empírico relativo ao uso de sistemas interativos.

Segundo [Rocha e Baranauskas 2003], usabilidade é um termo bastante amplo que se refere a quão fácil é para o usuário final aprender a usar um sistema, quão eficientemente ele irá utilizar um sistema assim que aprender a utilizá-lo e quão agradável é o seu uso.

O *design* da interface de um sistema é uma atividade que requer análise dos requisitos dos usuários, concepção, especificação e prototipação da interface, e por fim a avaliação da utilização do protótipo pelos usuários [de Souza et al. 1999].

Para simular a interação do usuário com o sistema que será desenvolvido, o projetista necessita utilizar modelos de interação, que descreverá de forma abstrata como o usuário pode interagir com o sistema.

Essa descrição abstrata oferece a vantagem de ser independente de estilo e ferramenta de interface [de Souza et al. 1999].

Ainda de acordo com [de Souza et al. 1999], o processo de design de interface segue o modelo da Figura 2.6. O desenvolvimento do sistema passa pela execução cíclica de tarefas. A cada ciclo novas modificações são feitas a fim de que o sistema seja aperfeiçoado. Cada ciclo envolve a especificação das funcionalidades e do modelo de interação, a prototipação de interfaces e a avaliação junto aos usuários. Na avaliação é verificada a existência de problemas, ou novos requisitos a serem atendidos, novo ciclo de desenvolvimento acontece, passando novamente pelas fases de especificação, prototipação e avaliação.



Figura 2.6: Processo de design de interfaces [de Souza et al. 1999]

De acordo com [Silveira 2002] deve ser feita a elicitación e análise de:

- Usuários
- Ambiente de trabalho dos usuários
- Tarefas dos usuários

Após a definição de quem são os usuários, qual o seu ambiente de trabalho e quais as tarefas que o sistema deve realizar, pode ser feita a modelagem da interação, que especifica os requisitos que um sistema deve possuir quanto a sua interface.

2.4.1 Métodos de Modelagem de Interação

A especificação de um sistema indica como ele funciona, sem se preocupar como ele será construído.

De acordo com [Silva 2007], a especificação de uma interface de usuário indica quais requisitos ela deve atender quanto à interação do usuário com o sistema, sem entrar em detalhes de implementação. Uma especificação de interface de usuário pode ser realizada utilizando linguagem natural ou através de linguagens ou notações específicas.

Alguns métodos e técnicas de modelagem de interação são [Silva 2007, Silveira 2002]:

- **Técnicas Baseadas em Gramática** As técnicas baseadas em gramáticas utilizam um conjunto de regras que definem a sintaxe de uma linguagem determinada, dentre elas destacam-se:

- *Backus-Naur Form (BNF)*

A BNF é uma técnica que utiliza operadores, nomes terminais e não-terminais. Os terminais representam as atividades do usuário na interface. Os não-terminais são uma combinação de terminais e não-terminais com operadores aritméticos e lógicos. Através de regras, que é a definição de um não-terminal, são definidas as interações dos usuários com o sistema [Silva 2007].

- *Task-Action Grammars (TAG)*

A TAG é uma variação da BNF. Utilizam regras gramaticais parametrizadas, que caracterizam as ações do usuário e descrevem a interação do usuário com o sistema [Payne e Green 1989].

- *Command Language Grammar (CLG)* A CLG provê um framework que é constituído de um conjunto de descrições, cada descrição é um nível diferente de abstração [Moran 1981].

- **Diagramas de transição de estados**

As técnicas baseadas em diagramas de transição de estados demonstram a interação do usuário com o sistema de forma gráfica. Dentre elas destacam-se:

- *State Transition Diagrams (STD)*

O STD é uma representação por meio de um grafo dos estados de um sistema através de nós, no qual cada nó é um estado.

Cada transição entre nós representa uma ação ou evento. Quando uma transição ocorre, o sistema executa uma ação, alterando assim o estado do sistema [Parnas 1969].

- *Statechart*

Os *statecharts* estendem o formalismo do STD, utilizando uma estrutura hierárquica, que se constitui de um diagrama simples para adicionar estrutura e mostrar estados alternativos e atividades concorrentes [Harel 1987].

- **Modelos Cognitivos**

Nos modelos cognitivos, são criados modelos mentais de como deve ser a interação do usuário com o sistema. O projetista deve identificar o usuário de acordo com suas particularidades, como por exemplo, conhecimento, cultura, experiência etc. Dentre elas destacam-se:

- **Goals, Operators, Methods and Selection rules (GOMS)**

A **GOMS** se baseia no trabalho sobre metas (**G**oals), operadores (**O**perators), métodos (**M**ethods) e regras de seleção (**S**election rules).

As metas representam o que o usuário deseja atingir, os operadores indicam as ações que o usuário executa para alcançar a meta desejada, um método representa a sequência de passos para executar uma tarefa e por fim a regra de seleção determina, dentre os métodos existentes para se alcançar uma meta, qual é o mais apropriado [Kieras 1999].

- **User Action Notation (UAN)**

A UAN possui uma notação que é orientada a usuário e tarefa descrevendo o comportamento do usuário e da interface e descrevendo como eles podem realizar uma tarefa de forma conjunta [Hartson et al. 1990, Hix e Hartson. 1993].

- **Linguagens de descrição de interfaces baseadas em *eXtensible Markup Language* (XML)**

[Silva 2007] afirma que recentemente, pesquisadores e analistas, entre outros, têm utilizado a XML para o desenvolvimento de linguagens de descrição de interfaces.

A XML é uma linguagem que representa dados através de tags que podem ser definidas pelo autor. Por conta dessa flexibilidade, é uma linguagem extensível, no qual novas tags podem ser criadas. Dentre elas destacam-se:

- **Abstract User Interface Markup Language (AUIML)** A AUIML é um vocabulário XML que foi desenvolvido para permitir a especificação de uma interação do usuário com um sistema.

O foco da AUIML não é a aparência da interface e sim a interação do usuário com o sistema independente de dispositivo de onde o sistema estará sendo executado [Azevedo et al. 2000].

- *User Interface Markup Language (UIML)*

A UIML é uma linguagem baseada em XML, a sua meta é permitir expressar interfaces de usuários para múltiplas plataformas de software em diferentes dispositivos e para múltiplas aplicações [Phanouriou 2000].

- *eXtensible interface Markup Language (XIML)*

A XIML é um framework para a definição de elementos de interação e seu relacionamento, pode representar itens de interface abstratos, concretos e relacionais.

- *USer Interface eXtensible Markup Language (USIXML)*

A USIXML descreve interfaces de usuários em vários níveis de detalhamento e abstração [Limboung et al. 2004].

- **Unified Modelling Language (UML)**

A UML é considerada a linguagem padrão para a especificação, visualização e documentação de software.

Existem extensões da UML para a modelagem de interface de sistemas computacionais, tais como a UMLi [Silva e Paton 2003].

- *UMLi (Unified Modelling Language for Interactive Applications)* [Silva e Paton 2000]

A UMLi é uma extensão da UML para modelagem de interface, que utiliza elementos dos casos de uso que fazem referência a interface e uma extensão da notação do diagrama de atividade para descrever interação e objetos do domínio da interface da aplicação [Silva e Paton 2000].

- *MOLIC (Modelling Language for Interaction as Conversation)*

A MOLIC é uma linguagem utilizada para modelar interação do usuário com os sistemas computacionais. Essa linguagem segue a metáfora da interação como conversa, e é fundamentada na engenharia semiótica [de Paula 2003].

2.4.2 Métodos de Inspeção de Usabilidade

Para encontrar problemas no design de uma interface é feito uma inspeção de usabilidade, que objetiva encontrar problemas e fazer recomendações para eliminá-los e melhorar a usabilidade do design [Rocha e Baranauskas 2003].

Dentre os métodos de inspeção pode-se destacar [Rocha e Baranauskas 2003]:

- **Avaliação Heurística**

A avaliação heurística é um tipo de inspeção no qual a avaliação é feita com base em uma lista de heurísticas;

- **Revisão de Guidelines**

A interface é avaliada utilizando uma lista de guidelines e verificando a conformidade da interface com cada item da lista.

- **Inspeção de Consistência**

O avaliador verificará se o sistema está em conformidade em relação a uma família de interfaces, verificando, por exemplo, se a terminologia, cores, layout estão de acordo.

- **Percurso Cognitivo**

Na avaliação por percurso cognitivo o avaliador simula o usuário “caminhando” na interface para realizar tarefas simples.

Para avaliar um sistema pode-se analisar as características da interface de acordo com as regras de ouro de [Shneiderman 1998] e as heurísticas de Nielsen [Nielsen 1993]. As 8 regras de [Shneiderman 1998] são:

1. **Esforçar-se pela consistência**

Terminologias, menus e cores devem sempre indicar a mesma coisa dentro de todo o sistema;

2. **Permitir que os usuários possam usar atalhos**

A interface deve prover aos usuários modos de acesso mais rápido, como uso de teclas especiais;

3. **Oferecer *feedback* informativo**

Toda ação do utilizador deve gerar *feedback*;

4. **Projeto de diálogos com princípio, meio e fim**

As sequências de ações devem estar organizadas em grupos;

5. **Oferecer prevenção e recuperação de erros**

O sistema deve evitar situações de erro, e de instabilidade;

6. **Permitir desfazer facilmente as operações**

As ações devem ser reversíveis

7. **Favorecer a sensação de controle**

Evitar situações que obriguem o usuário a iniciar ações em vez de simplesmente responder;

8. **Reduzir a carga na memória de curta duração**

Simplificar o projeto, consolidar múltiplas janelas, reduzir o movimento entre janelas.

Segundo [Nielsen 1993], dez regras podem ser usadas para auxiliar na avaliação de um sistema:

1. Visibilidade do status do sistema

A interface deve informar ao usuário o que está acontecendo, fornecendo *feedback* instantâneo para orientá-lo;

2. Relacionamento entre a interface do sistema e o mundo real

A interface deverá ser coerente com o que usuário já conhece;

3. Liberdade e controle do usuário

Deve permitir ao usuário desfazer ou refazer ações. Poder voltar para um ponto onde estava, quando estiver perdido em situações inesperadas;

4. Consistência

Efetuar operações, e usar os mesmos termos o tempo todo, evitando fazer a mesma ação com ícones e palavras diferentes. Elementos similares devem ser tratadas de forma similar;

5. Prevenção de erros

Ações como exclusões de itens, ou solicitações devem sempre vir acompanhadas com diálogos para confirmar a certeza em executar uma ação, evitando assim erros;

6. Reconhecimento ao invés de lembrança

Evitar acionar a memória do usuário, interface deve oferecer ajuda contextual, o sistema deve dialogar com o usuário;

7. Flexibilidade e eficiência de uso

O sistema deve ser fácil para o uso por usuários leigos, e ao mesmo tempo flexível para se tornar ágil para usuários avançados;

8. Estética e design minimalista

Diálogos devem ser simples e diretos, presentes somente quando necessário. Evitar excessos na comunicação;

9. Ajude os usuários a reconhecer, diagnosticar e sanar erros

As mensagens de erro não podem intimidar o usuário, devem indicar solução para o erro, ou uma saída;

10. Ajuda e documentação

Um bom design evita ao máximo a necessidade de ajuda na utilização do sistema. Mas deve existir para orientar o usuário.

2.5 Lógica Proposicional

Lógica está relacionada com raciocínio e com a validade de argumentos [Coppin 2010]. Na lógica a preocupação não é a veracidade de sentenças e sim a validade delas. Para que se possa analisar uma situação é importante a criação de argumentos sobre ela. Considere-se o argumento do Exemplo 4.

Exemplo 4 - Se estivesse chovendo e Jane não estivesse com seu guarda-chuva, então ela se molharia. Jane não está molhada. Está chovendo. *Portanto*, Jane está com seu guarda-chuva [Huth e Ryan 2008].

O argumento é válido se a frase seguida por **portanto** for uma consequência lógica das anteriores.

A linguagem da lógica proposicional é baseada em proposições ou frases declarativas. Uma proposição é uma sentença declarativa que pode ser interpretada como verdadeira ou falsa [Souza 2008].

Considere o argumento:

1. Se está chovendo e Jane não está com um guarda-chuva então ela se molha
2. Está chovendo e Jane está sem um guarda-chuva
3. Portanto, Jane se molha

Escrevendo de maneira mais formal os argumentos acima podem ser escritos como:

1. Se $\check{P} \wedge \check{Q}$ então $\check{R}$
2. $\check{P} \wedge \neg \check{Q}$
3. Portanto, $\check{R}$

Com a utilização dos símbolos proposicionais $\check{P}$, $\check{Q}$ e $\check{R}$, a preocupação não é o significado do que $\check{P}$, $\check{Q}$ ou $\check{R}$ representam, mas sim a forma da fórmula. A ênfase não é o estudo de argumentos expressos em linguagem natural, mas sim o estudo da forma dos argumentos que são representados pelos símbolos proposicionais [Souza 2008].

Para a definição da linguagem da Lógica Proposicional, seguem-se os passos: definição do alfabeto da linguagem e definição das regras gramaticais para a construção das proposições a partir do alfabeto [Souza 2008]

Definição 6: Alfabeto da Lógica Proposicional

- Símbolos de pontuação: $(,)$;
- Símbolos de verdade: true, false;
- Símbolos proposicionais: $P, Q, R, S, P_1, Q_1, R_1, S_1, P_2, \dots$;
- Conectivos proposicionais: $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$.

2.5.1 Síntese dos Conectivos da Lógica Proposicional

Dadas as seguintes frases atômicas:

P: Está chovendo

Q: Jane está com o guarda-chuva

R: Jane se molha **Sintaxe do conectivo proposicional $\neg$ (negação)**

A negação de Q é denotada por $\neg Q$, e expressa “Jane não está com o guarda-chuva” o que é equivalente a “não é verdade que Jane está com o guarda-chuva”. **Sintaxe do**

conectivo proposicional $\wedge$ (conjunção)

Sentenças que possuem a palavra **e** em Português, para expressar mais de um conceito, podem ser traduzidas para a lógica utilizando o operador **E**, $\wedge$.

Assim a expressão “Está chovendo e Jane não está com o guarda-chuva” pode ser denotada por $P \wedge \neg Q$, conhecido como conjunção de P e $\neg Q$. **Sintaxe do conectivo**

proposicional $\vee$ (disjunção)

Sentenças que possuem a palavra **ou** em Português, para expressar conceitos disjuntos, podem ser traduzidas para a lógica utilizando o operador **OU**, $\vee$.

Assim a expressão “Jane está com o guarda-chuva ou Jane se molha” pode ser denotada por $Q \vee R$, conhecido como disjunção de Q e R . **Sintaxe do conectivo proposicional**

$\rightarrow$ (implicação)

Sentenças que possuem a palavras **se** seguida de **então** em Português, para expressar condição ou implicação, denota que a frase atômica que se segue da palavra então é consequência lógica da frase atômica que se segue da palavra se. Assim a expressão “Se está chovendo e Jane não está com um guarda-chuva então ela se molha” pode ser denotada por $(P \wedge \neg Q) \rightarrow R$

Sintaxe do conectivo proposicional $\leftrightarrow$ (bi-implicação)

Sentenças que possuem a palavras **se e somente se** em Português, para expressar uma bi-implicação, denota que as frases atômicas que veem antes e depois das palavras se e somente se são consequências lógicas uma da outra. Assim a expressão “Jane se molha se e somente se está chovendo” pode ser denotada por $R \leftrightarrow P$.

A cada objeto sintático, como as proposições P e Q , pode ser associado um significado ou interpretação. Como dito acima, uma proposição é uma sentença declarativa que pode ser interpretada como verdadeira ou falsa. Assim se P é uma proposição, logo ela pode ser interpretada como verdadeira ou falsa.

A semântica de $P \wedge Q$, é definido como “verdadeiro quando P é verdadeiro e Q é verdadeiro” [Coppin 2010]. Assim podem-se interpretar sentenças lógicas ignorando o que os símbolos proposicionais significam, interpretando assim a forma com que os símbolos estão representados nas sentenças.

A seguir, indicam-se algumas definições importantes para o estudo da semântica da lógica proposicional, segundo [Souza 2008].

Definição 7: Fórmulas

As formulas são construídas a partir dos símbolos do alfabeto, conforme as regras a seguir:

- Todo símbolo de verdade é uma fórmula;
- Todo símbolo proposicional é uma fórmula;
- Se H é uma fórmula, então $(\neg H)$, a negação de H , é uma fórmula;
- Se H e G são fórmulas, então a disjunção de H e G , dada por: $H \vee G$, é uma fórmula;
- Se H e G são fórmulas, então a conjunção de H e G , dada por: $H \wedge G$ é uma fórmula;
- Se H e G são fórmulas então a implicação de H e G , dada por: $(H \rightarrow G)$ é uma fórmula. Nesse caso H é o antecedente e G é consequente da fórmula $(H \rightarrow G)$;
- Se H e G são fórmulas, então a bi-implicação de H e G , dada por: $(H \leftrightarrow G)$ é uma fórmula. Nesse caso H é o lado esquerdo e G é o lado direito da fórmula $(H \leftrightarrow G)$.

Definição 8: Função Binária

Uma função é binária se seu contradomínio possui apenas dois elementos

Definição 9: Função Total

Uma função é total se é definida em todos os elementos de seu domínio.

Definição 10: Função Interpretação

Uma interpretação I , na Lógica Proposicional, é uma função binária total na qual:

- O domínio de I é constituído pelo conjunto das fórmulas da Lógica Proposicional;
- O contradomínio de I é o conjunto $\{T, F\}$.

2.5.2 Semântica dos conectivos da Lógica Proposicional

Pela **Definição 10**, uma interpretação I é uma função total, significando que está em todos os elementos de seu domínio. Dado um símbolo $\check{P}$, que representa qualquer símbolo proposicional, então ele está contido no domínio de I . Assim podem ocorrer duas possibilidades, $I[\check{P}] = T$ ou $I[\check{P}] = F$ [Souza 2008].

O conjunto de valores lógicos contém dois elementos, T e F , em que T representa “verdadeiro” e F representa “falso” [Huth e Ryan 2008].

Semântica do conectivo proposicional $\neg$ (negação)

$I[\neg P] = F$ se $I[P] = T$, ou seja, se $I[P] = T$ então $I[\neg P] = F$

Semântica do conectivo proposicional $\wedge$ (conjunção)

Tabela 2.3: Tabela-verdade da fórmula $P \wedge Q$.

$I = [P]$	$I = [Q]$	$I = [P \wedge Q]$
T	T	T
T	F	F
F	T	F
F	F	F

Tabela 2.4: Tabela-verdade da fórmula $P \vee Q$.

$I = [P]$	$I = [Q]$	$I = [P \vee Q]$
T	T	T
T	F	T
F	T	T
F	F	F

Tabela 2.5: Tabela-verdade da fórmula $P \rightarrow Q$.

$I = [P]$	$I = [Q]$	$I = [P \rightarrow Q]$
T	T	T
T	F	F
F	T	T
F	F	T

Tabela 2.6: Tabela-verdade da fórmula $P \leftrightarrow Q$.

$I = [P]$	$I = [Q]$	$I = [P \leftrightarrow Q]$
T	T	T
T	F	F
F	T	F
F	F	T

A interpretação do conectivo $\wedge$ é o da conjunção de proposições. As possíveis interpretações da sentença $P \wedge Q$ são dadas pela terceira coluna da tabela-verdade da Tabela 2.3.

Semântica do conectivo proposicional $\vee$ (disjunção)

A interpretação do conectivo $\vee$ é o da disjunção de proposições. As possíveis interpretações da sentença $P \vee Q$ são dadas pela terceira coluna da tabela-verdade da Tabela 2.4.

Semântica do conectivo proposicional $\rightarrow$ (implicação)

A interpretação do conectivo $\rightarrow$ é o da implicação de proposições. As possíveis interpretações da sentença $P \rightarrow Q$ são dadas pela terceira coluna da tabela-verdade da Tabela 2.5.

Semântica do conectivo proposicional $\leftrightarrow$ (bi-implicação)

A interpretação do conectivo $\leftrightarrow$ é o da bi-implicação de proposições. As possíveis interpretações da sentença $P \leftrightarrow Q$ são dadas pela terceira coluna da tabela-verdade da Tabela 2.6.

Capítulo 3

Uma Proposta para Elicitação de Requisitos pelo Usuário: Árvore de Características

Este capítulo apresenta uma proposta para elicitação de requisitos através de um modelo gráfico de fácil desenvolvimento e compreensão. O estudo e os resultados da aplicação desse modelo foram publicados em [Oliveira et al. 2010d].

No desenvolvimento de um sistema, é imprescindível o conhecimento das necessidades que os usuários possuem. O objetivo é definir formas de elicitação de requisitos que sejam simples até mesmo para usuários leigos que desejem definir as necessidades de um software.

No caso deste trabalho, o método de modelagem é exemplificada na elicitação dos requisitos de uma comunidade virtual de aprendizagem. E, nesse caso, o exemplo contou com o apoio de professores do ensino fundamental para realizar a referida elicitação de requisitos.

Em geral, o cliente ou usuário leigo não consegue especificar claramente tudo que precisa, deixando para o desenvolvedor a tarefa de prever e imaginar suas necessidades. E ao fazer isso o cliente acata tudo que foi sugerido pelo desenvolvedor, para no futuro apontar os problemas, ou, eventualmente, questionar os requisitos que devem ser reavaliados e refeitos. Tudo isso, torna o processo de elicitação de requisitos demorado e com alto custo.

Nesta proposta, é utilizado um modelo gráfico para descobrir e anotar as características que o sistema deve possuir. Entretanto, é importante observar que esse processo de descoberta dos requisitos não é um processo matemático e há fatores organizacionais, técnicos e sociais envolvidos [Batista 2003].

Os desenvolvedores de software têm procurado a melhor forma de levantar os requisitos de um sistema e, dessa forma, diversas técnicas e métodos foram criados para garantir que

um sistema atenda às necessidades e expectativas dos clientes [Batista 2003, De Bortoli e Price 2000, Neto et al. 2003, Pressman 2006]

No processo de elicitação de requisitos, algumas técnicas são empregadas: entrevistas, questionários e *brainstorming*. [Pressman 2006]. Além disso, de uma forma geral, os sistemas são projetados seguindo proposições *ad hoc* de especialistas [Neto et al. 2003]. Neste trabalho, é proposta uma variante dessas técnicas, a qual utiliza um modelo gráfico que concentra os requisitos do sistema, em vários níveis de abstração. Tal modelagem, baseada na metodologia de desenvolvimento guiado por característica, apresenta as seguintes características:

- Pode ser elaborada por usuários que não são especialistas, possibilitando o desenvolvimento por qualquer usuário, a fim de recolher requisitos necessários em um sistema, organizando-os em uma árvore de características;
- É um modelo que estimula o design participativo e envolve os clientes, ou usuários, em todo o processo de desenvolvimento de software [Dix 2004].

Essa abordagem prevê que desenvolvedor e cliente trabalhem em um modelo gráfico simples e fácil de compreender. Isso facilita a coleta de requisitos e melhora a qualidade do software final, pois fica mais claro o entendimento do sistema solicitado. Há, também, um incremento da satisfação do cliente ao receber um software que se adequa às suas necessidades.

3.1 Árvore de Características

Para a definição dos requisitos de um sistema existem várias técnicas tais como a elaboração de diagramas de casos de uso e diagrama de fluxo de dados. São diagramas cujo desenvolvimento é feito por especialistas. Porém, quem conhece o que o software deverá prover é o usuário final do sistema.

Dessa forma, é necessário que o cliente de um software tenha uma ferramenta para auxiliar o projetista de um sistema na definição das características que um software deve possuir. Todos os diagramas utilizados para definir os requisitos de um software, no entanto, necessitam de conhecimento técnico, o que não é viável para a utilização pelo usuário leigo nas técnicas de engenharia de software para elicitação de requisitos.

Neste trabalho foi desenvolvido o conceito de árvore de característica, que se constitui em uma representação gráfica dos requisitos de um sistema através de uma árvore.

Na formação dessa árvore, o usuário é impelido, pelo tipo de modelagem, a detalhar cada vez mais o modelo, pois partindo de uma característica genérica o modelo em árvore impulsiona o usuário a detalhar cada uma das características, assim formando uma árvore que procura ser completa.

Para exemplificar o uso de árvores de características é apresentada, a seguir, a modelagem de uma comunidade virtual de aprendizagem com foco na educação de crianças.

A perspectiva atual da educação passou a ser apoiada por novas tecnologias, ambientes e sistemas de software, das quais podem se destacar as comunidades virtuais de aprendizagem, tais como os ambientes Moodle¹ e Teleduc². As comunidades virtuais de aprendizagem têm incorporado cada vez mais recursos para o desenvolvimento colaborativo de conteúdo educacional, como por exemplo, wikis e blogs. O que significa que o processo de aprendizado é melhor quando os alunos têm um perfil mais próximo de produtores de conhecimento do que de mero consumidores [Alves 2006].

Uma proposta de comunidade virtual educacional voltada para crianças foi publicada em [Oliveira et al. 2010a], seu desenvolvimento foi baseado nos modelos descritos nesta dissertação, bem como em outros trabalhos publicados pelos autores [Oliveira et al. 2010c, Oliveira et al. 2010b].

3.2 Modelando uma comunidade virtual de aprendizagem

Define-se, a seguir, um metamodelo de ensino no qual são delineadas suas principais funcionalidades. Nesse contexto, cada funcionalidade é composta por características específicas.

A Figura 3.1 apresenta um exemplo de aplicação da proposta. Tem-se, nesse caso, um metamodelo de ensino de lógica. Observa-se que a escolha do assunto não é determinante na modelagem e poderia ser outro assunto qualquer.

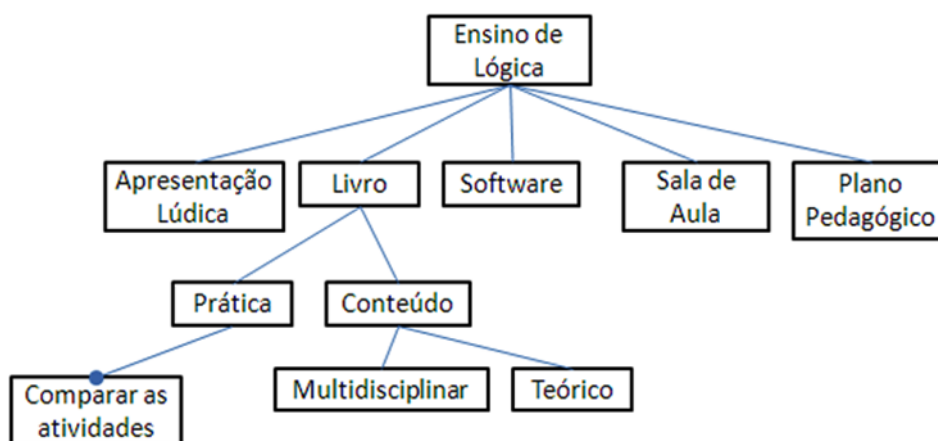


Figura 3.1: Parte do modelo específico inicial para ensino de lógica à distância

Para a criação da árvore, são consideradas algumas características fundamentais do problema:

¹<http://moodle.org>

²<http://www.teleduc.org.br>

Apresentação Lúdica

A forma de ensinar deve ser lúdica, pois os alunos são crianças do 6º ao 9º ano do ensino fundamental, que são atraídas por ambientes interativos e divertidos.

Livro

Todo o ensino é baseado em um livro texto, específico, que possui conteúdo teórico e multidisciplinar. No caso do ensino de lógica, o projeto utiliza como texto básico a série “Belisca no Mundo da Lógica” [Souza 2009].

Software

O ensino à distância necessita de um software para apoiar a aprendizagem e facilitar a interação e colaboração.

Sala de Aula

Pode existir uma sala de aula para um ensino coletivo do assunto.

Plano Pedagógico

É necessário um plano pedagógico que detalhe a metodologia do ensino e objetivos.

A partir do ponto no qual se definem as características principais, cada uma delas é expandida para que se possa olhar para as características de forma mais detalhada. Definem-se, então, os requisitos filhos, que detalham as características que a definem determinada pelo nó anterior.

A característica *Software*, por exemplo, indicada na Figura 3.1, é mostrada na Figura 3.2 como Portal de Ensino. Ela foi expandida para agregar as características que a compõem.

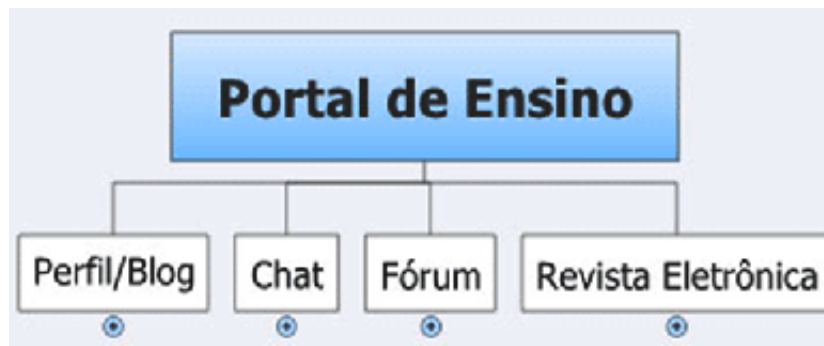


Figura 3.2: Modelo do “Portal de Ensino”

No modelo da Figura 3.2, foram identificadas as seguintes características do Portal de Ensino:

- **Perfil/Blog**

Na comunidade virtual, cada aluno possui um perfil e um blog. Essa característica é composta de outras que estão especificadas na Figura 3.3(a).

- **Chat**

Permite uma comunicação síncrona entre os alunos, ou entre o professor, o tutor e os alunos.

• Fórum

Ferramenta de comunicação assíncrona que permite o envio e resposta de dúvidas, ou mensagens de assuntos gerais. As características fórum e chat são compostas de outras, e estas podem ser visualizadas na Figura 3.3(b).

• Revista Eletrônica

Essa característica concentra o ensino do material didático, no âmbito do projeto é baseado no livro “Belisca no Mundo da Lógica” [Souza 2009]. Trata-se de um livro eletrônico interativo, utilizado como facilitador do ensino de lógica. As características que a compõe podem ser visualizadas na Figura 3.3(c).

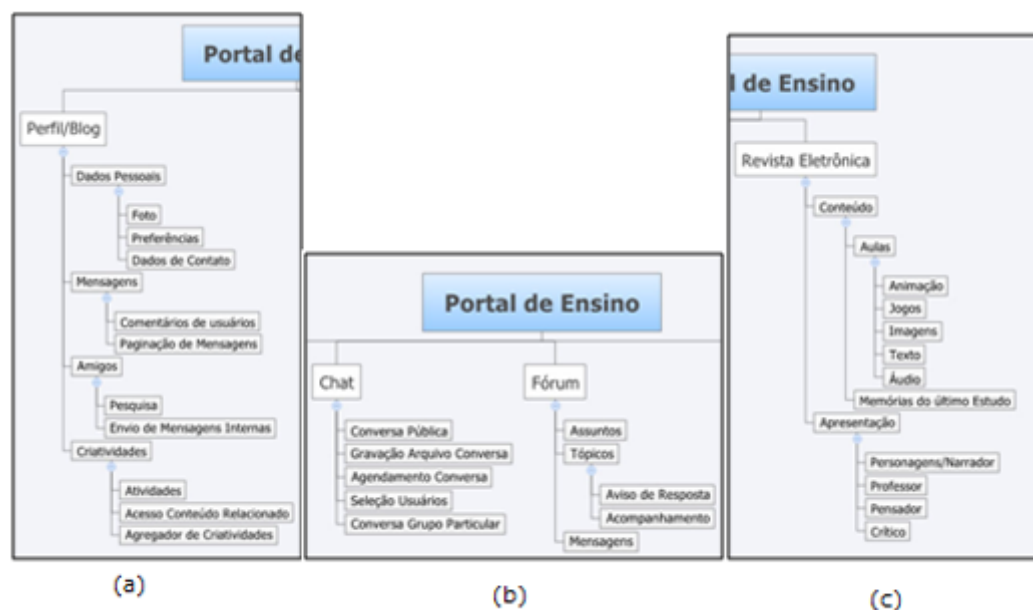


Figura 3.3: Modelos das características

Como foge do escopo deste trabalho se aprofundar em o que é e como se compõe uma comunidade virtual de aprendizagem, não são considerados os detalhes de cada característica definida.

Esse tipo de modelagem estimula o pensamento criativo, um *brainstorming* pessoal. Isso porque, combinando e estendendo ideias, é possível focar, individualmente, nas características e expandí-las.

De acordo com [Osborn 1962] *brainstorming* é um processo criativo, uma técnica de criatividade, que se dedica a combinar e estender ideias, incentivar ideias pouco convencionais e com foco na quantidade. Combinam frases, palavras relacionadas a um tema, para assim gerar soluções. Com a combinação dessas ideias, ou seja, características de um software, cria-se um modelo similar a uma árvore, a que denominamos árvore de características, para futuras referências.

O objetivo dessa modelagem é estabelecer um conjunto de procedimentos simples que podem ser executados por qualquer cliente em conjunto com o desenvolvedor do sistema. Assim, o modelo tem como característica a sua simplicidade, o que o torna adequado para ser elaborado pelo cliente em conjunto com o desenvolvedor.

Nesse contexto, as características são apresentadas de forma estruturada. Sendo os requisitos do sistema, em particular, organizados de forma hierárquica.

Além disso, os requisitos são detalhados ao longo da construção da árvore, o que facilita a execução das outras etapas da engenharia de requisitos: análise, especificação e validação [Batista 2003].

3.3 Experimentação e Resultados

A fim de comprovar a simplicidade na aplicação desta abordagem com usuários leigos, foi realizada uma experimentação com 44 professores da rede pública de ensino. Foi solicitado aos participantes do experimento, que desenvolvessem uma árvore de características para uma comunidade virtual de aprendizagem.

O objetivo era avaliar a facilidade dos professores na elicitação de requisitos, seguindo a metodologia do desenvolvimento da árvore de característica.

Para o desenvolvimento dessa atividade alguns passos foram seguidos:

- Em um primeiro momento foi apresentado aos professores os conceitos a respeito do que é a Internet, Web 2.0 e comunidades virtuais de aprendizagem.
- Foram explicados, brevemente, os fundamentos da elaboração de uma árvore de características, exemplificando os conceitos com a modelagem de uma loja virtual de presentes. A Figura 3.4 mostra a árvore de característica de uma loja virtual.

Esse modelo, em outro domínio, foi utilizado para que não houvesse a influência de um modelo já pronto, sobre os participantes.

- Após esse breve exemplo, foram distribuídos os questionários com uma folha anexa para a criação da árvore.

3.3.1 Perfil dos Participantes

Como o objetivo do método proposto neste trabalho é facilitar a elaboração de sistemas que atendam usuários leigos, foi feita uma pesquisa sobre o perfil dos participantes do experimento.

A maioria dos participantes já utiliza a Web há bastante tempo (25 de 44 participantes), porém o conhecimento em informática é mediano.

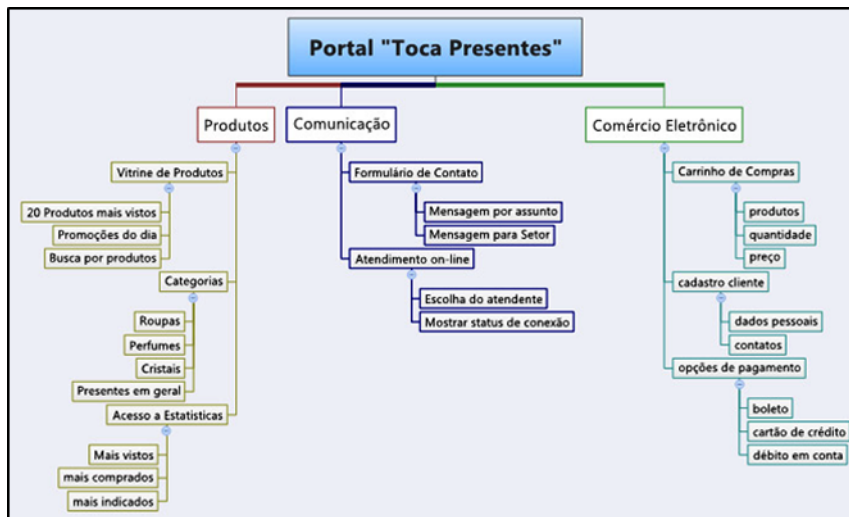


Figura 3.4: Exemplo de uma árvore de características de uma loja virtual.

No que se refere à área de atuação dos professores participantes, a maioria se concentrava no ensino fundamental. Alguns eram professores do Ensino Médio e da Educação de Jovens e Adultos. Além disso, havia também alguns que eram coordenadores de área e laboratoristas.

3.3.2 Análise dos Resultados

Durante o experimento, cerca de 20% dos participantes solicitaram auxílio para sanar dúvidas, ou mesmo para que fossem feitas verificações em seus modelos, dado o desejo de certificarem se estavam procedendo de forma correta.

Alguns participantes compararam o modelo a mapas mentais, ou mesmo a organogramas, o que, para eles, facilitou a elaboração da árvore. Esses participantes ainda comentaram que se sentiam impulsionados a detalhar cada vez mais o nó antecedente, o que permitia uma análise mais refinada do sistema.

De um modo geral, a ideia foi captada por todos e houve um entendimento da metodologia. Além disso, todos os participantes do experimento foram capazes de produzir árvores de características em graus variados de detalhamento.

Houve casos em que alguns adaptaram o modelo com características específicas de sua área de atuação, como foi o caso de uma professora de inglês de turmas de 8º e 9º ano do ensino fundamental, cuja árvore é mostrada na Figura 3.5.

A Figura 3.6 apresenta uma árvore que contém várias características para uma comunidade virtual de aprendizagem genérica desenvolvida por uma professora de educação física do ensino fundamental.

Como é o objetivo do sistema proposto, as modelagens foram realizadas de forma isolada, sem o acompanhamento de desenvolvedores. Entretanto, mesmo encontrando dificuldades, apenas duas pessoas não se sentiram capacitadas para completar a atividade. Os demais realizaram a atividade com graus variados de detalhamento.

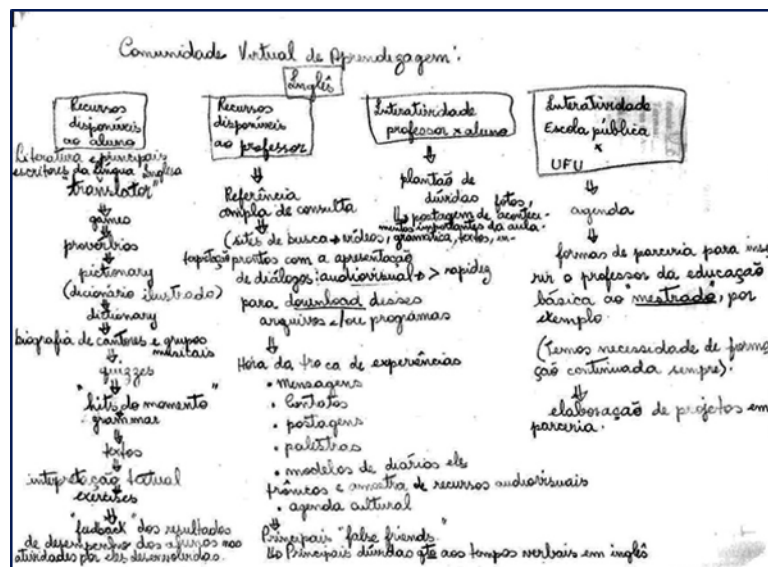


Figura 3.5: Ambiente Educacional por uma professora de Inglês.

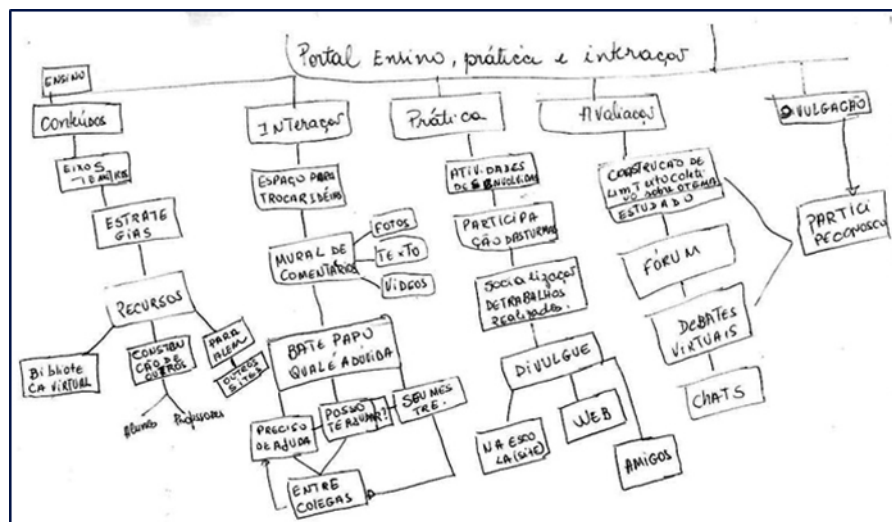


Figura 3.6: Árvore genérica por uma professora de Educação Física.

A Figura 3.7 apresenta o nível de complexidade, ou seja, o nível de dificuldade que os participantes do experimento encontram na elaboração da modelagem.

Cada participante avaliou a sua dificuldade e a pontuou em um nível de 1 a 10. Observa-se que mais de 50% dos participantes classificaram sua dificuldade em um nível igual ou inferior a cinco.

3.3.3 Comentários dos participantes

A seguir são transcritos alguns comentários feitos pelos participantes, quando questionados do porquê de terem definido certo nível de dificuldade para realizar a modelagem.

São apresentados, inicialmente, os comentários de alguns participantes que definiram o nível de dificuldade igual ou inferior a cinco:

"Por que através do conhecimento da realidade de minha escola não é possível montar

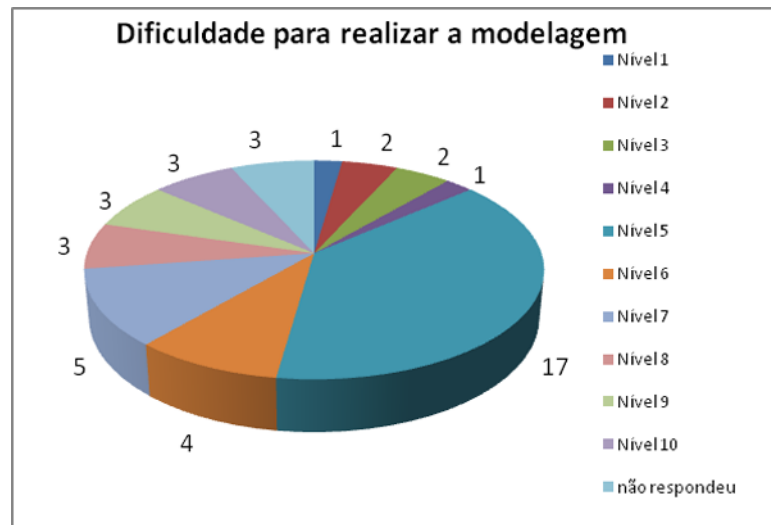


Figura 3.7: Níveis de dificuldade encontrados para realizar a modelagem.

um sistema on-line que vá atender todas as necessidades. Lembrando que o mesmo estará sempre em processo de construção conforme a necessidade do momento.”

“No princípio, foi por falta de entender o que foi pedido. Ao esclarecer o que era para ser feito, ficou fácil.”

Comentários de usuários que definiram o nível de dificuldade superior a 5:

“Nós não estamos acostumados a pensar o processo de criação. Apenas repetimos, reproduzimos e utilizamos o que o outro criou. Criar e inventar requer hábito.”

“Acho que pela minha falta de experiência na informática. Tenho vontade de aprender, gosto de aplicar em sala de aula, mas reconheço minhas limitações.”

Em geral, a dificuldade de alguns reside principalmente na falta de conhecimentos de informática, ou mesmo da falta de pensamento criativo. Isso, porque durante o exercícios tiveram que pensar como colaboradores e não como meramente usuários finais.

O experimento propiciou a retirada dos indivíduos de uma posição a que estavam acostumados, como consumidores, para então pensarem como provedores de recursos.

3.3.4 Incorporação dos resultados

Após analisar cada árvore desenvolvida, foi possível incorporar algumas características em uma árvore. Alguns passos foram realizados para a incorporação dos resultados:

- Requisitos similares são agrupados;
- Os requisitos cujos escopos não estão de acordo com o sistema são eliminados;
- Os requisitos que não estão em um nível técnico condizente são eliminados.

A Figura 3.8 apresenta a árvore resultante da incorporação das características apontadas pelos participantes como importantes para uma comunidade virtual de aprendizagem.

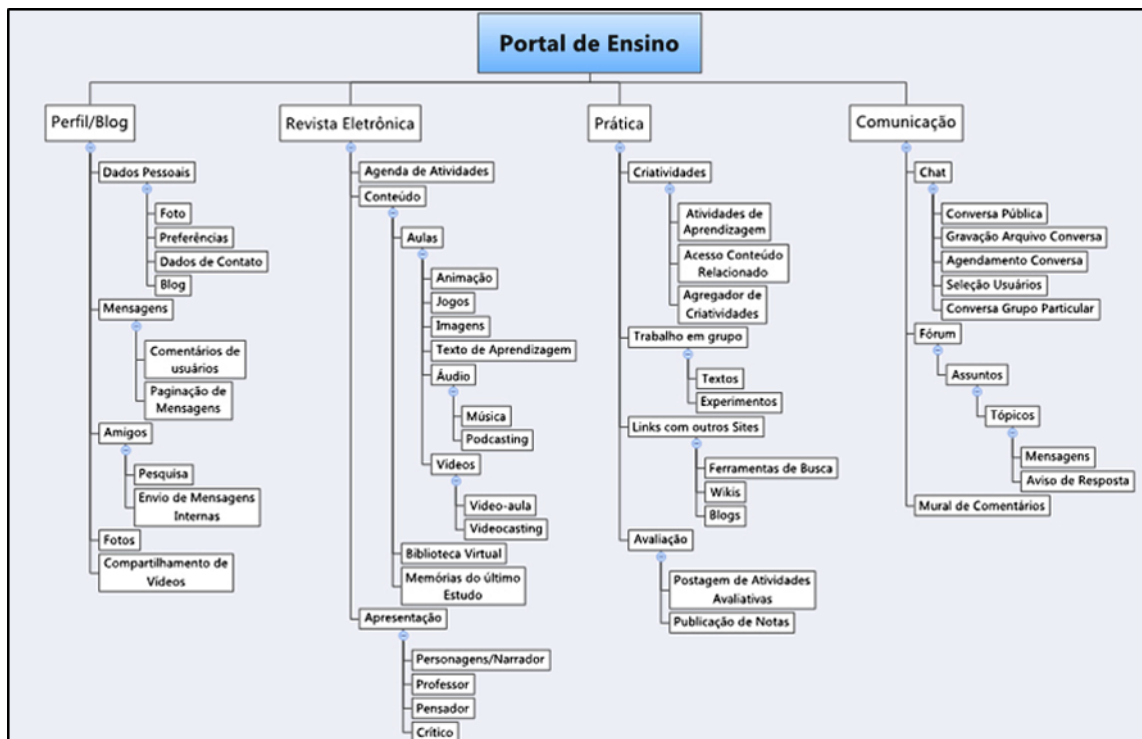


Figura 3.8: Árvore de características resultante [Oliveira et al. 2010d].

Capítulo 4

Rede de Petri Colorida com expressões da Lógica Proposicional: Modelagem de Fluxo de Navegação em Interfaces de Software

No desenvolvimento de uma interface gráfica de, por exemplo, uma Comunidade Virtual de Aprendizagem é necessário um grupo multidisciplinar. Um grupo que contenha, pelo menos, engenheiros de software, pedagogos, *designers* gráficos, etc. E para que esse grupo funcione sob as mesmas necessidades do software e possa se comunicar eficazmente, o uso de um Roteiro de Atividades é, em geral, utilizado em alguns casos [Souza et al. 2007]. Mas, geralmente, o Roteiro de Atividades é um documento extenso em linguagem natural, sujeito a erros e no qual, facilmente, algumas características do software podem não ser bem detalhadas.

Além do Roteiro de Atividades, há outras formas de contemplar a comunicação eficaz do grupo desenvolvedor de software.

Um desses modelos é, por exemplo, o modelo gráfico para a elicitação de requisitos de um software. Nesse modelo, proposto em [Oliveira et al. 2010d], o objetivo principal é a boa compreensão, por todo o grupo de trabalho, das necessidades do sistema. Em [Oliveira et al. 2010d], procura-se atender esse objetivo, utilizando árvores de características.

Neste trabalho, sugerem-se também as árvores de características como ferramenta no processo de modelagem de interface e de fluxo de navegação (Figura 3.8). Nesse caso, a modelagem de árvore de características é utilizada para o levantamento das necessidades, com a ajuda dos usuários leigos que são conhecedores das necessidades do sistema.

Em seguida, tendo sido levantadas as necessidades do sistema, o modelo formal proposto utilizará uma extensão da Rede de Petri Colorida para modelar a interface e fluxo de navegação do sistema.

Como parte do estudo, um levantamento de outros possíveis modelos também é feito. Em uma análise comparativa dos modelos, o modelo proposto é, então, refinado.

4.1 Levantamento de Necessidades em um Processo de Design de Interface Gráfica

Para ilustrar o processo de levantamento das necessidades, na modelagem de uma interface gráfica e seu fluxo de navegação, este trabalho utilizando árvores de características, como propostas em [Oliveira et al. 2010d]. Nesse caso, as árvores de características levantam as necessidades e características de uma Comunidade Virtual de Aprendizagem, ou portal de ensino (Figura 3.8).

A Figura 3.8 foi desenvolvida através da mesma pesquisa com 44 professores da rede municipal de ensino da cidade de Uberlândia-MG descrita no capítulo anterior. Objetivou-se a geração de árvores de características que correspondessem ao que os professores viam como necessidades para o desenvolvimento e uso de uma comunidade virtual de ensino.

A partir da geração dessas árvores as características foram unificadas em uma única estrutura, que é apresentada na Figura 3.8. As características se constituem em requisitos de sistemas, e esses requisitos podem ser de natureza variada, como por exemplo, requisitos funcionais, não funcionais (utilidade, confiança, desempenho, suporte e escalabilidade) [Pressman 2006].

A árvore de característica da Figura 3.8 apresenta alguns dos requisitos que foram identificados para uma comunidade virtual de ensino. Ela apresenta na raiz da estrutura a identificação do sistema, em seguida esse nó é expandido nas características vistas como principais, como por exemplo, a presença de um perfil de usuário, área de blog, revista eletrônica que concentra a parte de conteúdo das disciplinas, atividades de aprendizagem e por fim a área de interação do portal. Cada ramo dessa árvore que parte do nó raiz irá fornecer características mais detalhadas das funcionalidades, auxiliando no projeto e desenvolvimento do sistema.

Observa-se que a modelagem de requisitos, a partir de uma árvore de características, é uma estratégia para a elicitação de requisitos efetuada pelo usuário que geralmente não possuem o conhecimento técnico dos projetistas de sistemas. Entretanto, tais usuários são os detentores do conhecimento a respeito do sistema que deve ser desenvolvido.

A árvore de característica resultante do processo de elicitação de requisitos não é, necessariamente, perfeita, pois ela reflete, unicamente, o que o sistema deve possuir sem a preocupação técnica de como ele deve ser desenvolvido.

No levantamento das necessidades de um sistema interativo, é necessário, inicialmente, saber quais são os requisitos do sistema. Nessa proposta, esse levantamento é obtido pela construção de uma árvore de características. Nela são expostas todas as necessidades do

sistema como, por exemplo, a identificação das telas e funcionalidades do software.

4.2 Grafo de Telas

Após o levantamento das necessidades do sistema e sua representação em uma árvore de característica, tal árvore é transformada em um grafo de telas.

O objetivo do grafo de telas é identificar, pontualmente, as macro telas do sistema, para que em seguida possam ser representadas utilizando a Rede de Petri.

Nessa tradução, da árvore de características em um grafo de telas, é necessária uma série de passos, entre os quais:

1. Detalhamento do modelo pelos projetistas do sistema (modificação da árvore de características para melhor compreensão dos engenheiros de software);
2. Realimentação do modelo pelos usuários (adição de detalhes, pelos usuários, desconsiderados inicialmente, reformulação de quesitos, etc.);
3. Retirar folhas e nós da árvore de característica que não representem requisitos funcionais;
4. Definir o Grafo de Telas;
5. Representar o Grafo de Telas, utilizando Rede de Petri.

Os dois primeiros passos são naturais na maioria das modelagens, nas quais projetistas e usuários trabalham em conjunto, em um sistema de auto realimentação. Os três últimos passos correspondem à tradução da árvore de características no grafo de telas.

- Retirar folhas e nós da árvore de requisitos que não representam requisitos funcionais

Inicialmente, a partir da árvore de características, são identificados os nós que podem ser representados por telas do sistema. Nesse caso, são excluídos os nós que representam requisitos não funcionais.

Na Figura 3.7, apresenta-se o “corte” das características que não serão utilizadas no grafo de telas.

Mesmo desconsiderando alguns nós, é importante enfatizar que a árvore de característica constitui importante apoio na modelagem do sistema. Isso porque nela estão declaradas as características que o sistema final deve apresentar. Dessa forma, ela é também uma ferramenta de verificação de conformidade com os desejos do usuário final.

- Definição do Grafo de Telas

Após a retirada, da árvore de requisitos, de folhas e nós que não representam requisitos funcionais, o próximo passo é transformar a árvore resultante em um grafo de telas. Nessa operação, o projetista deve identificar quais são as macro telas do sistema e como elas se interconectam.

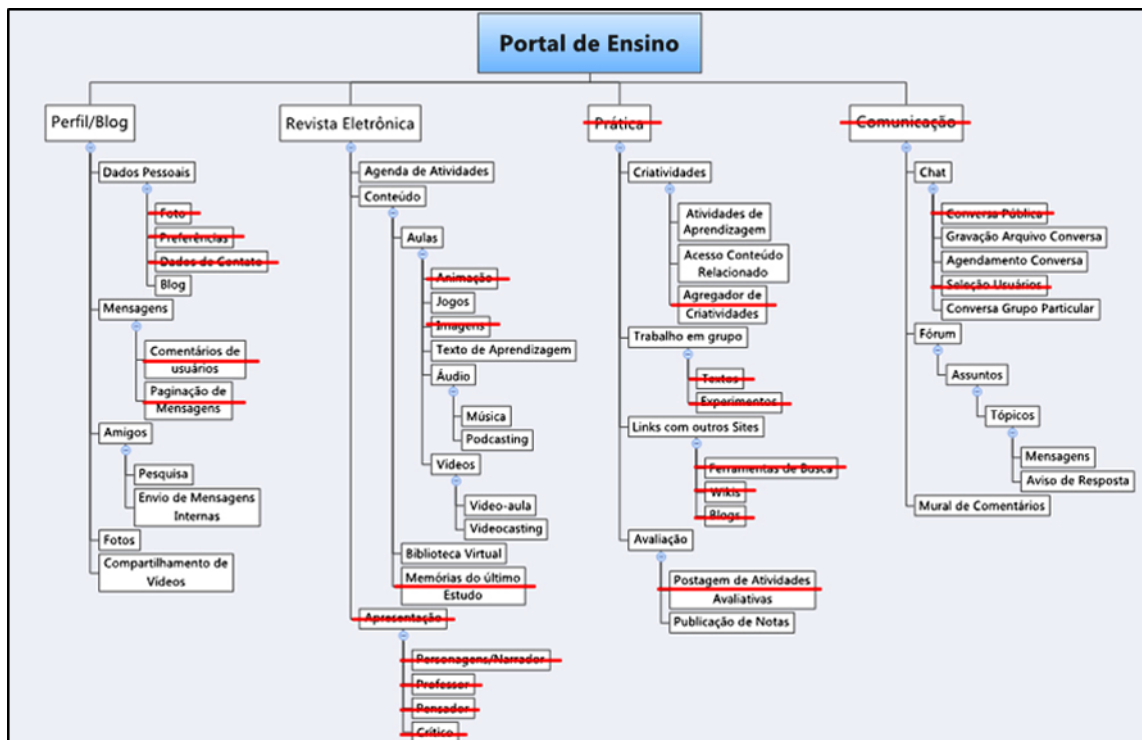


Figura 4.1: “Corte” das características que não constituem telas do sistema.

Nesse contexto, os nós do grafo representam as telas do sistema e seus arcos representam as possíveis transições entre as telas. Assim, os arcos que conectam os nós do grafo são bi-direcionados se a comunicação entre as telas são do tipo “ida e volta”. Além disso, no caso de um grafo conexo, por exemplo, todos os nós são alcançáveis a partir da tela principal (página ou tela inicial do sistema).

A Figura 4.2 apresenta o grafo de telas que foi desenvolvido a partir da seleção dos nós na árvore de características da Figura 3.8, que representam telas ou funcionalidades do Portal de Ensino.

Para o desenvolvimento do grafo de telas, inicialmente, é estabelecido um nó principal que representa a tela inicial do sistema. A partir desse nó são conectados com arcos outros nós que representam telas do software.

A ligação entre as telas deve ser analisada pelo projetista, considerando, nessa etapa, o fluxo entre as telas do sistema.

No exemplo da Figura 4.2, o nó *Página Principal* é a tela inicial do sistema.

A partir dela são alcançadas as telas *Perfil*, *Revista Eletrônica*, *Biblioteca Virtual*, *Criatividade*, *Chat*, *Fórum* e *Mural de Recados*.

A decisão de quais telas podem ser alcançadas a partir da tela inicial é um elemento de projeto. Porém, como indicativo de relevância da tela, os nós da Árvore de Características mais próximos da raiz, geralmente, são candidatos a nós filhos da tela inicial.

O link, por exemplo, entre as telas “Revista Eletrônica” e “Biblioteca Virtual” é uma decisão de projeto, mesmo pertencendo a uma ramificação diferente. Considerando fatos

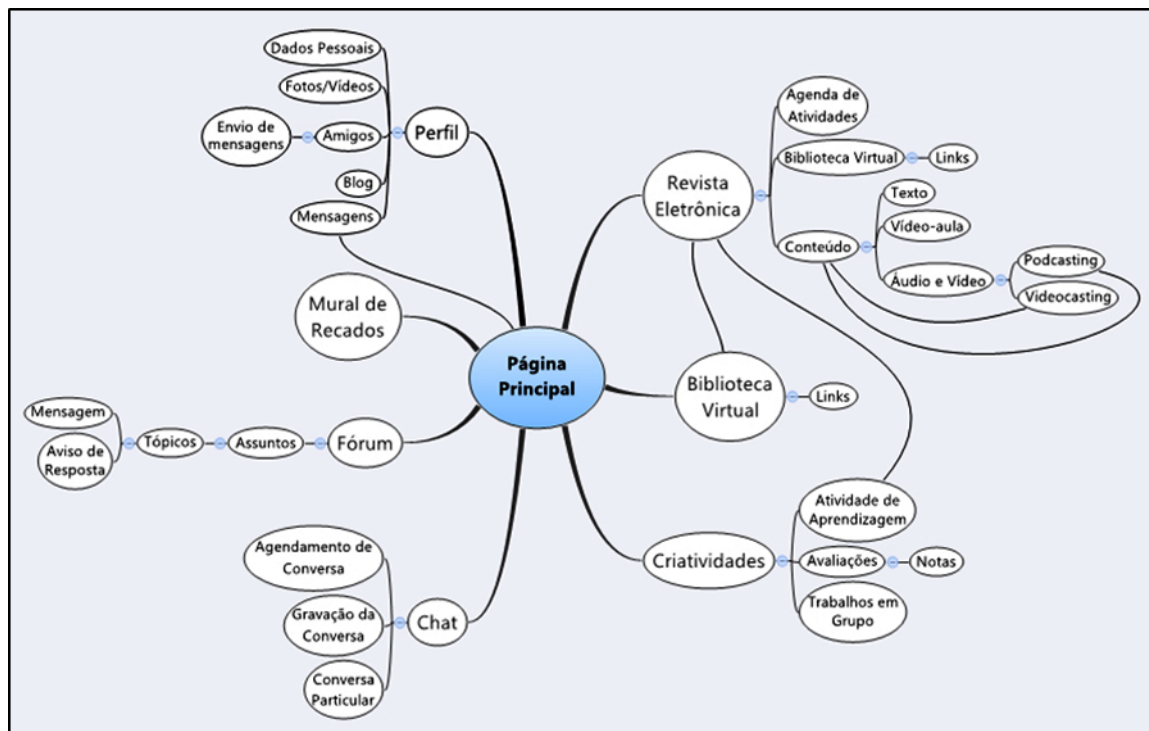


Figura 4.2: Grafo de Telas.

como esse, o projetista define como as telas do sistema se comunicam.

- Representação do grafo de telas, utilizando Rede de Petri.

O grafo de telas obtido no passo anterior é, então, representado, utilizando Redes de Petri. Nesse caso, pode ser utilizada a Rede de Petri ordinária ou suas extensões. Nesse trabalho é proposta a representação dos Grafos de Telas, utilizando Redes de Petri Coloridas com expressões da Lógica Proposicional.

4.3 Rede de Petri Colorida com expressões da Lógica Proposicional

A Rede de Petri Colorida com expressões da Lógica Proposicional, denotada por RdP-CeLP, é uma extensão da Rede de Petri Colorida, obtida pela adição de expressões lógicas associadas aos arcos e às transições.

A associação de uma linguagem da lógica às Redes de Petri se deve, fundamentalmente, aos três aspectos básicos da Lógica [Harel et al. 2000]:

- A presença de uma linguagem formada de expressões bem formadas, sobre as quais podem ser atribuídos significados. Têm-se os símbolos da linguagem, juntamente com conjuntos de regras formais para composição de expressões bem formadas. Isso difere de um agregado arbitrário de símbolos, sendo denominado de sintaxe da linguagem.

- O estudo da semântica associada à linguagem define como interpretar expressões bem formadas sobre um universo do discurso. Esse universo pode ser composto de objetos matemáticos tais como grupos, gráficos ou números naturais, ou elementos do cotidiano como automóveis, empregados ou o tempo. A declaração sobre a linguagem comunica a respeito das propriedades e relações entre os objetos.
- Associado à linguagem e à semântica, tem-se um sistema dedutivo que consiste de uma coleção de regras que podem ser aplicadas para derivar, de uma forma puramente mecânica, fatos interessantes a respeito da relação entre os objetos semânticos. Nesse contexto, os fatos e relacionamentos são expressos pela linguagem da Lógica.

Assim, para definir uma RdPCeLP é necessário, inicialmente, estabelecer a linguagem da Lógica Proposicional utilizada para definir as expressões lógicas que são associadas às transições e arcos da RdPCeLP. Em seguida, a semântica da linguagem é, também, definida. E, finalmente, o sistema mecânico de dedução formal associado é analisado.

Ou seja, na modelagem proposta neste trabalho, são seguidos os passos:

1. Especificação de uma linguagem baseada na Lógica Proposicional;
2. Definição da semântica associada à linguagem que define a RdPCeLP;
3. Definição do sistema formal de dedução associado à RdPCeLP.

4.3.1 Linguagem da Lógica Proposicional utilizada na RdPCeLP

Definição 11: Alfabeto

O alfabeto da Lógica Proposicional é constituído por:

- Símbolos de pontuação: (,);
- Conjunto enumerável de símbolos proposicionais para lugares $p, q, r, s, p_1, q_1, r_1, s_1, p_2, \dots$;
- Conectivos: $\neg, \wedge, \rightarrow$

Definição 12: Literal

Um literal é um símbolo proposicional ou sua negação. Portanto, um literal L é dado por:

$$L ::= p / \neg p$$

tal que p é um símbolo proposicional qualquer.

O símbolo proposicional p é um literal positivo e $\neg p$ é um literal negativo.

Exemplo 5 As expressões:

$$p, \neg q \text{ e } r$$

são literais.

Exemplo 6

A expressão

$$p \wedge \neg q \wedge r$$

é uma cláusula.

Neste exemplo, se

$$C = p \wedge \neg q \wedge r$$

então C representa a expressão da lógica proposicional na qual p , q e r estão relacionadas em uma operação de conjunção.

Definição 14: Subtração de cláusulas

Dadas duas cláusulas C_1 e C_2 , a subtração de C_2 em C_1 , denotada por $C_1 \setminus C_2$ é dada pela conjunção dos literais de C_1 menos os literais de C_2 .

Exemplo 7

Dado duas cláusulas C_1 e C_2 , se:

$$C_1 = p \wedge q \wedge \neg r$$

$$C_2 = p$$

Então:

$$C_1 \setminus C_2 = q \wedge \neg r$$

Definição 15: União disjunta:

Dadas duas cláusulas C_1 e C_2 , a união disjunta entre tais cláusulas, denotada por

$$C_1 \dot{\cup} C_2$$

é uma conjunção de literais. A cláusula $C_1 \dot{\cup} C_2$ é obtida considerando os casos:

Se C_1 ou C_2 contém o literal positivo p , então p é um literal da conjunção

$$C_1 \dot{\cup} C_2$$

Se C_1 contém o literal negativo p , então p não é um literal da conjunção

$$C_1 \dot{\cup} C_2$$

Exemplo 8

Dado duas cláusulas C_1 e C_2 , se:

$$C_1 = p \wedge \neg r$$

$$C_2 = q \wedge r$$

Então:

$$C_1 \dot{\cup} C_2 = p \wedge q$$

O operador de união comum, quando operando sobre duas cláusulas faria a conjunção indiscriminada dos literais. Porém quando uma cláusula possui um literal e a outra cláusula possui a negação desse mesmo literal, a atuação do operador de união comum consideraria ambos na união. Exemplificando:

$$C_1 = p \wedge \neg r$$

$$C_2 = q \wedge r$$

Então:

$$C_1 \dot{\cup} C_2 = p \wedge \neg r \wedge q \wedge r$$

A união comum dessas cláusulas não determinará assim a presença ou ausência de um símbolo proposicional em um lugar. Esse símbolo proposicional é similar à ficha colorida da Rede de Petri Colorida, ela simboliza uma informação, ou recurso em um lugar.

No modelo RdPCeLP, os literais das expressões da lógica representam as fichas coloridas que são analisadas e geradas pelas transições. E cada transição ou arco da rede é associada a uma expressão da Lógica Proposicional, conforme a definição a seguir.

Definição 16: Implicação

Uma implicação E , da Lógica Proposicional, é definida pela regra gramatical:

$$E ::= C_1 \rightarrow C_2$$

tal que C_1 e C_2 são cláusulas quaisquer.

Observe que no contexto da proposta deste trabalho, uma expressão é um caso particular de fórmula da Lógica Proposicional. A razão de tal restrição é analisada mais

adiante.

Exemplo 9

Seja,

$$C_1 = p \wedge \neg r$$

$$C_2 = q$$

Então:

$$C_1 \rightarrow C_2 = (p \wedge \neg r) \rightarrow q$$

4.3.2 Representação Gráfica da Rede de Petri Colorida com expressões da Lógica Proposicional

Uma Rede de Petri Colorida com expressões da Lógica Proposicional, RdPCeLP, é uma extensão da Rede de Petri Colorida. Tal rede é um grafo orientado, com uma marcação inicial indicada por M_0 .

Analogamente a uma Rede de Petri ordinária e Rede de Petri Colorida, a RdPCeLP possui dois tipos diferentes de nós, denominados por lugares e transições. Os arcos da rede, necessariamente, ligam lugares a transições, ou transições a lugares.

No caso da RdPCeLP, os lugares e transições podem possuir rótulos, que indicam a função desempenhada pelo elemento dentro da rede. Nesse caso, tais rótulos podem oferecer uma semântica ao modelo, ou serem utilizados para identificar o elemento na rede.

Portanto, uma RdPCeLP é um grafo rotulado que possui, basicamente, três elementos: lugares, transições e rótulos. Tais elementos são representados como se segue:

- A Figura 4.3(a) representa um lugar, denotado por um círculo e uma cláusula Lg_1 que descreve o seu conteúdo;
- A Figura 4.3(b) representa uma transição, denotada por um retângulo e rotulada pelo seu nome;
- A Figura 4.3(c) representa um arco rotulado pela expressão C_1 .

4.3.3 Semântica da Linguagem da Lógica Proposicional utilizada na RdPCeLP

A Rede de Petri Colorida com expressões Lógicas é uma extensão da Rede de Petri Colorida, obtida pela adição de expressões lógicas associadas aos lugares, arcos e as transições.

Semântica das cláusulas que representam lugares

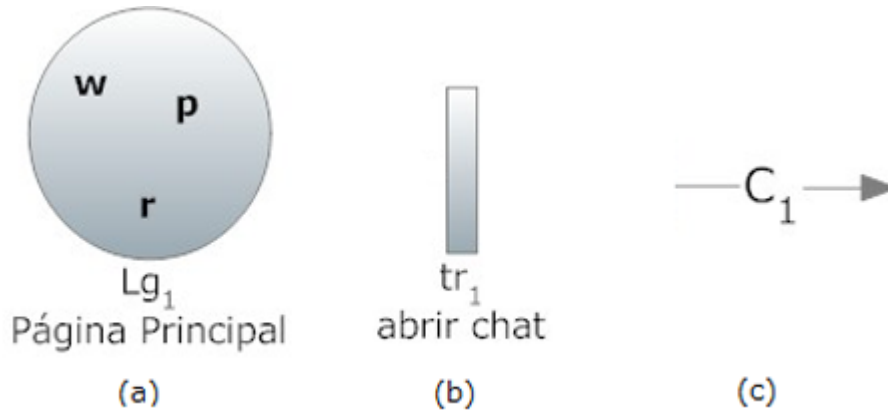


Figura 4.3: Representação gráfica dos nós da RdPCeLP.

Cada lugar é representado por uma cláusula, que corresponde a especificação das cores das fichas que o lugar contém.

Exemplo 10

Considerando o lugar indicado na Figura 4.4, nele há três fichas com as cores p, r e w. Nesse caso, o lugar é representado pela cláusula Lg_1 a seguir:

$$Lg_1 = w \wedge p \wedge r$$

Essa cláusula é dada pela conjunção dos literais que representam as fichas coloridas pertencentes ao lugar. O lugar Lg_1 é representado pela cláusula Lg_1 e desenhado conforme a Figura 4.4.

A representação gráfica de um lugar consiste no círculo com as fichas dispostas internamente.

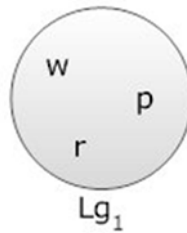


Figura 4.4: Lugar Lg_1 .

O lugar da Figura 4.5, Lg_2 , possui duas fichas coloridas, w e r . Assim, tem-se que:

$$Lg_1 = w \wedge r$$

é uma cláusula que representa esse lugar.

Os lugares Lg_1 e Lg_2 contêm apenas literais positivos. Mas, não necessariamente é isso que ocorre. Os lugares também contêm literais negativos.

Neste trabalho é considerada a seguinte convenção:

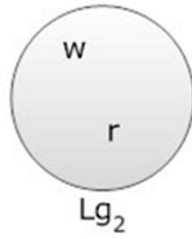


Figura 4.5: Lugar Lg2.

Convenção sobre os literais dos lugares

Se um lugar não contém o literal p , então ele contém $\neg p$. Um lugar não pode conter os literais p e $\neg p$.

Interpretação de um literal em um lugar

Seja L um literal e Lg um lugar, Lg satisfaz L se, e somente se, L ocorre, ou pertence ao lugar Lg .

Esse fato é denotado por:

$$I[L, Lg] = T$$

Interpretação de uma cláusula em um lugar

Seja C uma cláusula e Lg um lugar, Lg satisfaz C se, e somente se, os literais que compõem C são satisfeitos em Lg .

Esse fato é denotado por:

$$I[C, Lg] = T$$

Semântica das expressões que rotulam os arcos

Cada arco de entrada em uma transição é rotulado com uma expressão que corresponde às cores das fichas do lugar de origem de tal arco.

Cada arco de saída de uma transição é rotulado com uma expressão que corresponde à geração de fichas coloridas que são colocadas nos lugares de destino do arco, ou, ainda, a retirada de fichas dos lugares de destino.

Exemplo 11

Seja C_e uma cláusula da lógica proposicional, que rotula um arco de entrada em uma transição, tal que

$$C_e = p \wedge \neg q \wedge r$$

A Figura 4.6 representa o arco de entrada de uma transição rotulado por C_e . Considerando que a interpretação de C_e , no lugar de origem Lge , é verdadeira, ou seja,

$$I[C_e, Lge] = T.$$

Isso significa que no lugar de origem desse arco existem as fichas coloridas p e r e não existe a ficha q .

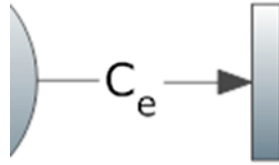


Figura 4.6: Arco de entrada rotulado pela cláusula C_e .

Seja C_s uma cláusula da lógica proposicional, que rotula um arco de saída de uma transição, tal que

$$C_s = q \wedge \neg r$$

Isso significa que se a transição for disparada, tem-se, no lugar de saída Lg_s que

$$I[C_s, Lg_s] = T.$$

Dessa forma, no lugar de destino Lg_s desse arco é gerada a ficha colorida q . Caso o lugar saída já tenha a ficha r , ela é retirada. Na Figura 4.7 está representado o arco de saída da transição rotulado pela cláusula C_s .

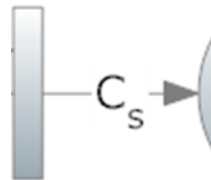


Figura 4.7: Arco de saída rotulado pela cláusula C_s .

Semântica das expressões das transições:

Na representação proposta, cada transição é associada a uma implicação lógica do tipo:

$$\{ \langle C_{e1}, L_{e1}, >, \dots, \langle C_{en}, L_{en}, > \} \rightarrow \{ \langle C_{s1}, L_{gs1}, >, \dots, \langle C_{sm}, L_{gs_m}, > \}$$

O antecedente dessa implicação corresponde às cláusulas que rotulam os arcos que chegam à referida transição e seus respectivos lugares de entrada.

O conseqüente corresponde às cláusulas que estão nos rótulos dos arcos que partem da transição e seus respectivos lugares de saída.

Nessa notação, a transição possui como lugares de entrada os lugares:

$$\{Lg_{e1}, \dots, Lg_{en}\}$$

que estão ligados à transição, respectivamente, por arcos rotulados pelas cláusulas:

$$\{Ce_1, \dots, Ce_n\}$$

Além disso, a transição possui como lugares de saída os lugares:

$$\{Lg_{s1}, \dots, Lg_{sn}\}$$

que estão ligados a transição, respectivamente, por arcos rotulados pelas cláusulas:

$$\{Cs_1, \dots, Cs_n\}$$

Na Figura 4.8 é apresentado um esquema geral da notação utilizada para as Redes de Petri Coloridas com expressões da Lógica Proposicional.

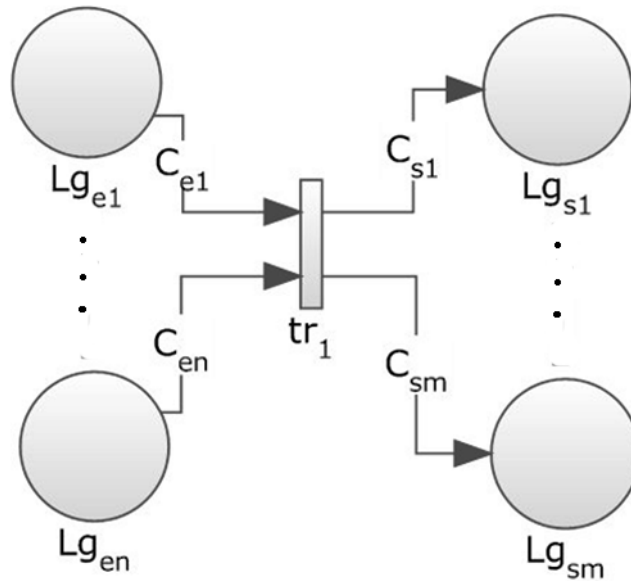


Figura 4.8: Esquema geral da Rede de Petri Colorida com Expressão Lógica Proposicional.

4.3.4 Condição de disparo da transição

Em uma RdPCeLP toda transição é rotulada por uma implicação do tipo:

$$\{ \langle Ce_1, Lge_1 \rangle, \dots, \langle Cen, Lgen \rangle \} \rightarrow \{ \langle Cs_1, Lgs_1 \rangle, \dots, \langle Csm, Lgsn \rangle \}$$

A condição do disparo da transição é dada pela análise do antecedente da implicação.

Se para todo i , $1 \leq i \leq n$, a cláusula de entrada, C_{ei} , é verdadeira no lugar de entrada Lg_{ei} , então a condição de disparo é satisfeita.

Portanto, se para todo i , $1 \leq i \leq n$,

$$I[C_{ei}, Lg_{ei}] = T.$$

então a condição de disparo é satisfeita.

Exemplo 12

Seja uma transição tr_1 que possua somente um arco de entrada que esteja rotulado pela cláusula C_1 ,

$$C_1 = p \wedge q \wedge \neg r$$

Considerando que tal transição parta de um lugar Lg que possui as fichas coloridas, p, q, w, p_1, q_3 e p_2 . Isso significa que Lg_1 é representado pela cláusula C_2 :

$$C_2 = p \wedge q \wedge w \wedge p_1 \wedge q_3 \wedge p_2$$

Esse lugar é representado na Figura 4.9.

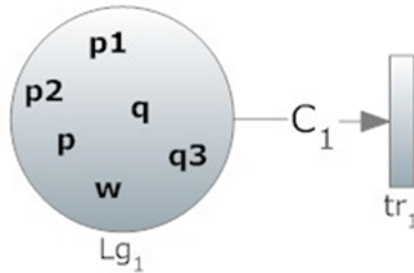


Figura 4.9: Arco de chegada em uma transição.

A condição de disparo da transição é dada por:

$$I[C_1, Lg_1] = T.$$

Ou seja, a interpretação de C_1 é verdadeira em Lg_1 .

Observando que isso somente ocorre se Lg_1 contém as fichas p, q e não contém a ficha colorida r .

4.3.5 Resultado do disparo de uma transição

Como resultado do disparo de uma transição tem-se dois casos:

- O primeiro caso é a geração de fichas, que ocorre de acordo com o rótulo dos arcos de saída da transição em seus respectivos lugares de destino.

- O segundo caso é o consumo das fichas relacionadas na cláusula que rotula o arco de entrada da transição. Nesse caso, esse consumo tem incidência sobre o lugar de origem do arco.

- Geração ou consumo de fichas coloridas nos lugares de destino

A geração ou consumo de fichas coloridas nos lugares de destino é obtida pela semântica do consequente da implicação que rotula a transição.

Considerando que a implicação a seguir rotule a transição t_r .

$$< Lg_1, C_{e1} > \rightarrow < Lg_2, C_2 >, < Lg_3, C_3 >$$

Nesse caso, a semântica relacionada ao disparo da transição é dada pelo consequente da implicação, que é dado por:

$$< Lg_2, C_2 >, < Lg_3, C_3 >$$

- Geração de fichas coloridas Considerando o disparo da transição tr_1 .

Nesse caso, todos os literais positivos da cláusula C_2 são adicionados ao lugar Lg_2 .

Da mesma forma, todos os literais positivos da cláusula C_3 são adicionados ao lugar Lg_3 .

Logo, se Lg_2 é a cláusula que representa Lg_2 , o lugar de saída é transformado em um novo lugar Lg'_2 tal que a nova cláusula que o representa é dada por Lg''_2 .

$$Lg''_2 = C_2 \dot{\cup} Lg'_2$$

- Consumo de fichas coloridas Considerando o disparo da transição tr_1 .

Nesse caso, suponha que C_2 contenha o literal negativo $\neg p$.

Se o lugar Lg_2 contém o literal p , então ele é retirado de Lg_2 .

Caso contrário, se Lg_2 não contém p , então Lg_2 continua não contendo p .

Logo, como no caso anterior, se Lg_2 é a cláusula que representa Lg_2 , o lugar de saída é transformado em um novo lugar Lg'_2 tal que a nova cláusula que o representa é dada por Lg'_2 .

$$Lg'_2 = Lg_2 \dot{\cup} C_2$$

Portanto, caso a cláusula que rotula um arco de saída da transição apresente um literal positivo, a ficha correspondente a esse literal é criada no lugar de destino, ou saída.

Caso a cláusula possua um literal negativo, então se o lugar de saída possuir a ficha referenciada pelo literal, ela é eliminada do lugar.

No efeito do disparo da transição tr_1 , os literais positivos da cláusula C_1 são adicionados ao lugar de saída.

Dessa forma, se C_2 representa o lugar de saída Lg_s antes do disparo de tr_1 , então após o disparo tr_1 , a cláusula C_2 é transformada em uma nova cláusula dada por C_3 tal que:

$$C_3 = C_2 \dot{\cup} C_1$$

Exemplo 13 Considera-se na Figura 4.10 um arco rotulado com C_1 , que parte de uma transição tr_1 e tem como destino um lugar Lg_1 , sendo que:

$$C_1 = q \wedge \neg r$$

$$Lg_1 = w \wedge r$$

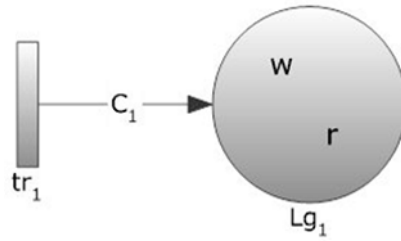


Figura 4.10: Disparo de uma transição.

Com o disparo de tr_1 , C_1 toma efeito sobre Lg_1 , ou seja, o literal positivo q será uma nova ficha colorida em Lg_1 , enquanto o literal negativo r irá retirar uma ficha colorida de Lg_1 pois ele possui uma ficha r .

Caso o lugar Lg_1 não possuísse uma ficha colorida r , o literal negativo da cláusula C_1 não iria causar nenhum efeito no lugar de destino.

“Consumo” de fichas coloridas nos lugares de origem

O consumo de fichas coloridas do lugar de origem dos arcos que entram na transição é dada pela análise do antecedente da seguinte implicação:

$$\langle Lg_1, C_1 \rangle \rightarrow \langle Lg_2, C_2 \rangle, \langle Lg_3, C_3 \rangle$$

Ou seja,

$$\langle Lg_1, C_{e1} \rangle$$

A semântica relacionada à essa relação expressa que a cláusula C_1 tem seus literais analisados e com efeito sobre Lg_1 .

Assim, se a cláusula possuir literais positivos esses irão consumir a ficha respectiva em Lg_1 .

Após C_{e1} ter sido analisada para o disparo da transição, as fichas coloridas são consumidas, seguindo a dinâmica similar que ocorre com as Redes de Petri ordinárias, nos

quais uma ficha é consumida do lugar de origem de um arco que chega em uma transição, sempre que essa transição for disparada.

Na Figura 4.11, temos que após o disparo da transição as cláusulas C_{ei} , $0 \leq i \leq n$, são analisadas. Nesses casos a cláusula que descreve Lg_i , sendo, $0 \leq i \leq n$, é modificada para Lg'_i tal que:

$$Lg'_i = Lg_i \setminus C_{ei} \quad \forall i$$

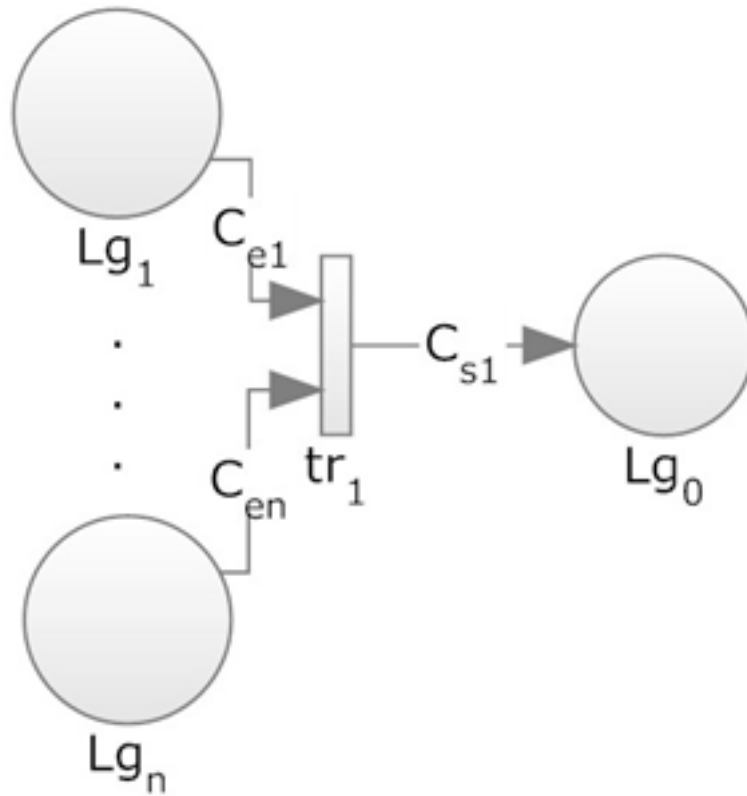


Figura 4.11: Exemplo de RdPCeLP.

Exemplo 14 Considera-se na Figura 4.12 um arco rotulado com C_1 , que entra em uma transição tr_1 e tem como origem um lugar Lg_1 , sendo que:

$$C_1 = p \wedge \neg q \wedge r$$

$$Lg_1 = p \wedge w \wedge r$$

Com o disparo de tr_1 , C_1 toma efeito sobre Lg_1 , ou seja, os literais positivos p e r farão com que ocorra a retirada das fichas coloridas p e r de Lg_1 . O literal negativo da cláusula não terá nenhum efeito sobre Lg_1 .

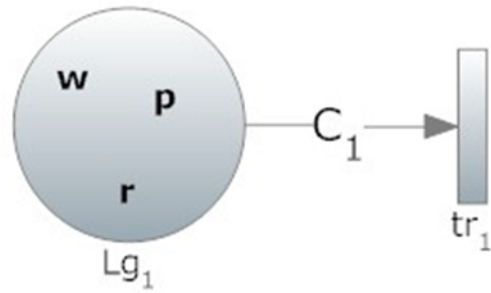


Figura 4.12: Arco rotulado com uma cláusula.

4.4 Exemplo de Rede de Petri Colorida com Expressão da Lógica Proposicional

A Rede de Petri da Figura 4.13 é um exemplo de Redes de Petri Coloridas com expressões da Lógica Proposicional. No qual, temos os lugares Lg_1 e Lg_2 , que são as cláusulas como segue:

$$Lg_1 = p \wedge w \wedge r$$

$$Lg_2 = w \wedge r$$

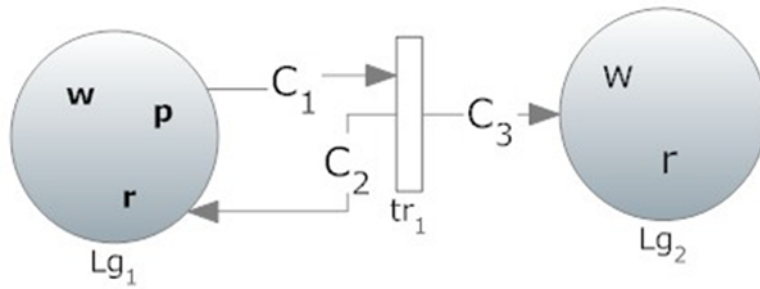


Figura 4.13: Exemplo de Rede de Petri Colorida com expressões Lógicas.

A Figura 4.13 possui ainda um arco de entrada da transição tr_1 e dois arcos de saída. Os três arcos são rotulados pelas cláusulas C_1, C_2 e C_3 , dos quais:

$$C_1 = p \wedge \neg q \wedge r$$

$$C_2 = \neg p$$

$$C_3 = q \wedge \neg r$$

A transição tr_1 seguirá um esquema do tipo:

$$< Lg_1, C_1 > \rightarrow < C_3, Lg_2 >, < C_2, Lg_1 >$$

O disparo de uma transição ocorre quando esta expressão da transição for satisfeita, ou seja, se tornar verdadeira. Isso significa que p e r estão presentes no lugar Lg_1 . Quando a transição é disparada a seguinte operação é efetuada:

- Interpretação de C_1 em Lg_1 com a retirada das fichas coloridas e a interpretação do arco de chegada rotulado por C_1 :

$$Lg'_1 = Lg_1 \dot{\cup} C_1$$

- Interpretação de C_3 em Lg_2 :

$$Lg'_2 = Lg_2 \dot{\cup} C_3$$

- Interpretação de C_2 em Lg_1 :

$$Lg''_1 = Lg'_1 \dot{\cup} C_2$$

Na expressão o efeito de C_2 em Lg_1 é nulo, pois a interpretação da retirada das fichas indicadas em C_1 já havia retirado a ficha colorida p .

Verifica-se que o modelo apresentado na Figura 4.13 satisfaz a cláusula C_1 , ou seja, $< L_1, C_1 >$, pois dado:

$$C_1 = p \wedge \neg q \wedge r$$

Tem-se que para $< Lg_1, C_1 >$ ser verdadeiro, a cláusula definida por $C_1 = p \wedge \neg q \wedge r$ deverá ser verdadeira, ou seja, deve existir no local Lg_1 os literais p e r , porém não deve existir q .

Assim caso o antecedente da implicação seja verdadeiro, o consequente acontece, ou seja, tendo $< Lg_1, C_1 > \rightarrow < C_2, Lg_2 >, < C_3, Lg_1 >$, e $< Lg_1, C_1 >$ é verdadeiro, então $< C_2, Lg_2 >, < C_3, Lg_1 >$ se sucede. Exemplifica-se o estado da RdPCeLP após o disparo da transição tr_1 da Figura 4.13 na Figura 4.14.

Assim após o disparo ocorre a retirada da ficha colorida p e r do lugar Lg_1 pois a interpretação de C_1 “consome” as fichas coloridas relacionadas em C_1 . Porém a cláusula C_2 não tem efeito sobre Lg_1 , pois esse já não possui a ficha colorida p .

$$C_1 = p \wedge \neg q \wedge r$$

$$C_2 = \neg p$$

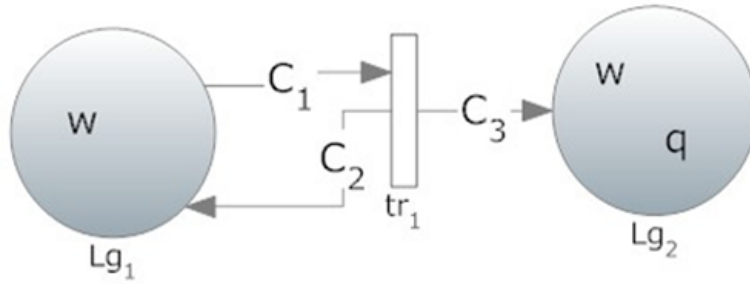


Figura 4.14: Modelo após o disparo de tr_1 .

Ocorre a colocação da ficha colorida q no lugar Lg_2 , e a retirada do símbolo r , o que corresponde à cláusula C_3 :

$$C_3 = q \wedge \neg r$$

Ambos os arcos C_2 e C_3 são as saídas da transição disparada, assim o que é expresso por eles é executado nos lugares de destino.

Caso, por exemplo, no lugar Lg_2 não houvesse um símbolo proposicional r , a única ação que aconteceria seria a colocação do símbolo q .

A semântica do modelo sugere então que os lugares Lg_1 e Lg_2 representam telas do sistema, e temos nelas fichas coloridas que representam funcionalidades da tela e a transição tr_1 corresponde a um programa que modifica as telas adicionando/habilitando ou desabilitando funcionalidades.

4.5 Definição Formal de RdPCeLP

Uma Rede de Petri Colorida com expressões da Lógica Proposicional, associada a uma marcação inicial, é uma quintupla dada por:

$$RdPCeLP = \langle P, T, Cor, Csc,$$

$M_0 \rangle$

na qual:

- P é um conjunto finito e enumerável de lugares, por exemplo telas/funções do sistema:

$$P = \{Lg_1, Lg_2, Lg_3, \dots, Lg_n\};$$

- T é um conjunto finito e enumerável de transições:

$$T = \{tr_1, tr_2, tr_3, \dots, tr_n\};$$

- Cor é um conjunto finito e enumerável de fichas coloridas:

$$Cor = \{a_1, b_1, c_1, a_2, \dots, a_n, b_n, c_n\};$$

- C é um conjunto finito de cláusulas que rotulam os arcos:

$$C = \{C_1, C_2, C_3, \dots, C_n\};$$

- C_{sc} é a função *subconjunto* de fichas coloridas que está associada a cada lugar e a cada transição, e é um subconjunto de C (cores possíveis para este lugar ou esta transição):

$$C_{sc} : P \cup T \rightarrow P(C_{or});$$

- W é a função de incidência:

$$W(p, t) : C_{sc}(t) \times C_{sc}(p) \rightarrow \mathbb{N};$$

- M_0 é a marcação inicial no qual a cada lugar um subconjunto de fichas coloridas:

$$M_0(p) : C_{sc}(p) \rightarrow \mathbb{N}$$

4.6 Estudo de Caso representado pela Rede de Petri Colorida com expressões da Lógica Proposicional

A seguir são apresentados exemplos de Rede de Petri com expressões da Lógica Proposicional (RdPCeLP) que modelam o fluxo de navegação e a interface de uma comunidade virtual de aprendizagem, foco de estudo desse trabalho.

Na Figura 4.15 apresenta-se a RdPCeLP, base do funcionamento de uma Comunidade Virtual de Aprendizagem. Essa Comunidade Virtual é um portal Web destinado a ser uma interface entre alunos e professores de um curso acadêmico.

O lugar principal, Lg1, rotulado por “Página Principal” apresenta todas as funcionalidades que podem ser acessadas na tela da página principal, que são representadas como fichas coloridas. São elas as ligações para os recursos de revista, criatividade, mural, biblioteca, mensagens, perfil, fórum e chat que a comunidade virtual de aprendizagem apresenta, e são representadas respectivamente pelas fichas coloridas *revista*, *criatividade*, *mural*, *biblioteca*, *mensagens*, *perfil*, *forum* e *chat*.

Cada arco que parte do nó “Página Principal” é rotulado por uma cláusula e ligado a uma transição.

A semântica relacionada a esse modelo indica que cada ficha colorida é um *link* dentro

da tela principal. Cada link ao ser clicado provocará uma mudança de tela. Isso, no modelo, corresponde à verificação da interpretação da cláusula. Caso essa interpretação seja verdadeira, ou seja, as fichas analisadas tornam a cláusula verdadeira, a transição será disparada.

No modelo da Figura 4.15, todas as transições estão “sensibilizadas”, ou seja, elas podem ser disparadas, pois todas as cláusulas que rotulam os arcos que partem da tela inicial possuem uma interpretação verdadeira.

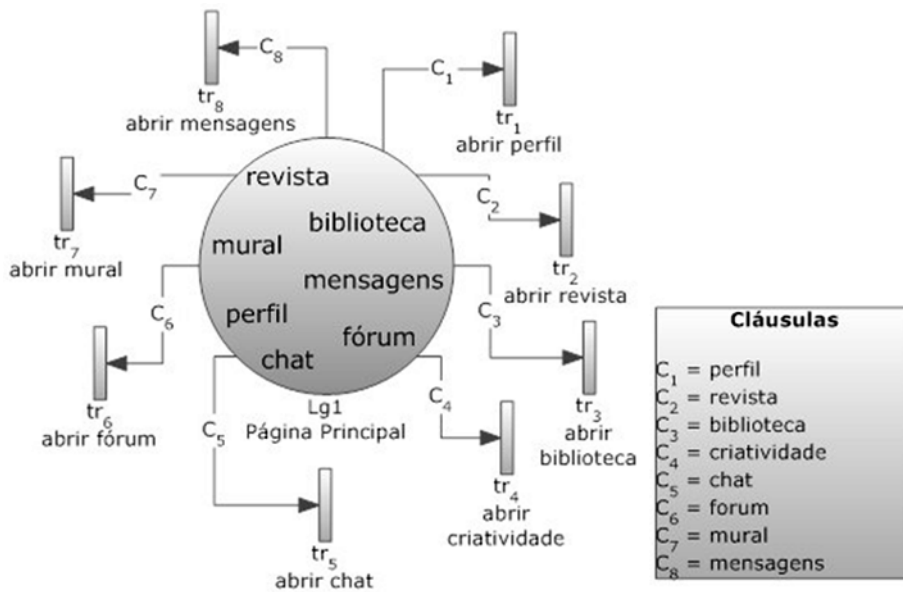


Figura 4.15: RdPCeLP da base da Comunidade Virtual de Aprendizagem.

O disparo de qualquer transição tr_i , com $1 \leq i \leq 8$, consumirá uma ficha do lugar Lg_1 , correspondente à cláusula C_i , com $1 \leq i \leq 8$, indicando que uma opção na Página Principal, assim que acessada, fará com que esta fique indisponível para ser aberta em uma segunda tela.

A Figura 4.15 representa uma rede principal, da qual partem as outras partes do sistema que foram suprimidas pois a rede é extensa.

Na Figura 4.16 é apresentada a modelagem da RdPCeLP do chat, acessada a partir do disparo da transição tr_5 da rede da Figura 4.15. A partir do modelo base da Figura 4.15, o disparo da transição rotulada por “abrir chat”, ou seja, a verificação da verdade da cláusula C_5 de acordo com sua interpretação em Lg_1 .

A mesma transição, “abrir chat”, está representada na Figura 4.16. Ao ser disparada, na Figura 4.15, ela irá “executar” a cláusula C_1 (Figura 4.16) no lugar Lg_1 rotulado por “chat”.

As cláusulas representadas na Figura 4.16:

$$C_1 = msg \wedge msgPriv \wedge agendaConversa \wedge registro \wedge historico \wedge \{amigo\}$$

$$C_2 = msg$$

$$C_3 = msgPrivada \wedge amigo$$

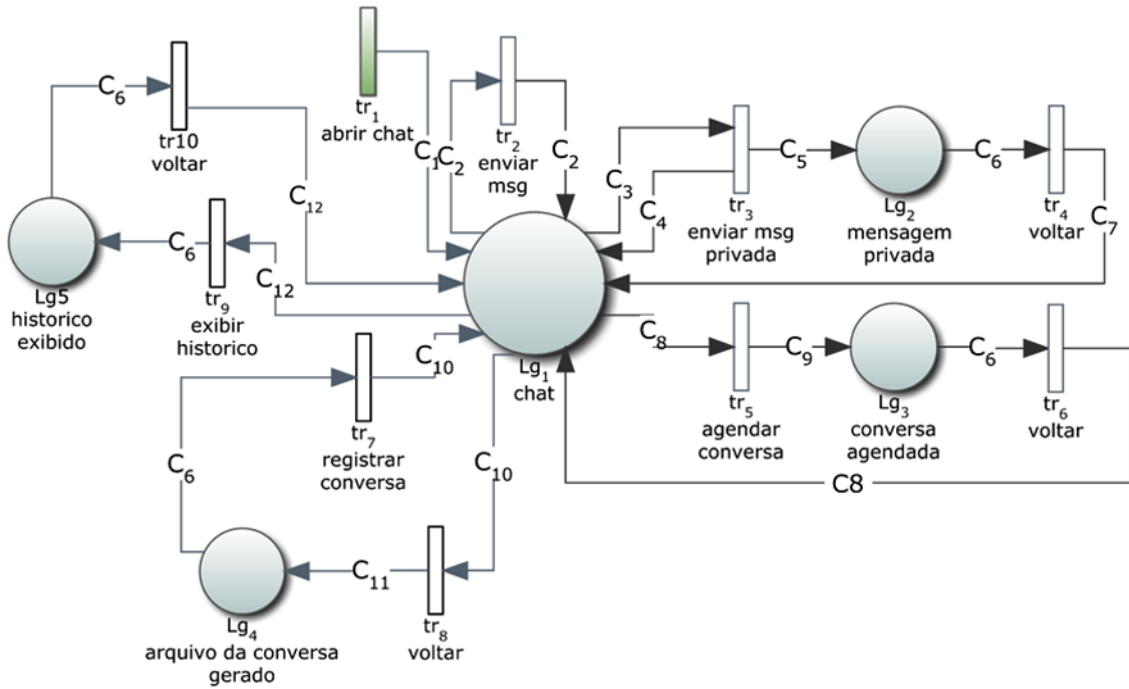


Figura 4.16: RdPCeLP de Chat.

$$C_4 = msgPrivada$$

$$C_5 = msgPrivada \wedge amigo \wedge voltar$$

$$C_6 = voltar$$

$$C_7 = amigo$$

$$C_8 = \{amigo\} \wedge agendaConversa$$

$$C_9 = \{amigo\} \wedge agendaConversa \wedge voltar$$

$$C_{10} = registro$$

$$C_{11} = msg \wedge voltar$$

$$C_{12} = historico$$

4.7 Propriedades da RdPCeLP

Algumas propriedades podem ser estudadas em relação à Rede de Petri Colorida com expressões Lógicas, especificamente aplicadas na modelagem do fluxo de navegação de sistemas computacionais.

Entre tais propriedades, têm-se as comportamentais, que correspondem às mesmas propriedades listadas para as Redes de Petri ordinárias. Porém, com a interpretação semântica relacionada ao escopo deste trabalho é aquela que considera o significado das propriedades da modelagem de fluxo de navegação. Evidentemente, uma modelagem que utiliza Redes de Petri Coloridas com expressões da Lógica Proposicional.

Essa semântica se relaciona principalmente às características que um sistema com interface de usuário deve possuir questões que se referem principalmente à usabilidade e

verificação de problemas na interface do sistema.

A seguir são listadas algumas das propriedades das Redes de Petri ordinárias, porém sob a ótica das RdPCeLP aplicadas na modelagem de fluxo de navegação de software:

- Alcançabilidade (*reachability*): Todas as telas projetadas para um sistema devem ser alcançáveis. Uma tela alcançável é um fator de avaliação do sistema, pois todas as telas projetadas devem possuir um meio de acesso a elas.
- Limitação (*boundedness*): O conceito de limitação se aplica a uma tela do sistema, caso tenhamos, por exemplo, um número limite de links de acesso em um menu de um site, ou mesmo, o número total de notícias em uma página não pode ultrapassar um total estipulado, etc., esse fator influencia o projeto do sistema. Tendo assim que ser avaliado para corresponder às necessidades do software.
- Vivacidade (*liveness*): A vivacidade é analisada na RdPCeLP em relação às situações nas quais o usuário pode chegar em uma tela e não ter meios de avançar para uma tela seguinte, ou mesmo retroceder para uma tela anterior.
- Reversibilidade (*reversibility*): Uma Rede de Petri é dita reversível caso a marcação inicial volte a ocorrer ao longo das sequências de disparo, assim na análise dos modelos de fluxo de navegação é desejável que o usuário possa retornar sempre para a tela inicial do sistema.
- Persistência (*persistence*): A garantia da propriedade de persistência no âmbito da modelagem de fluxo de navegação assegura que uma vez em uma tela o usuário pode optar por todas as opções disponíveis, sem que uma escolha afete o acesso à outra, sendo que ambas podem ser acessadas.

4.8 Análise das propriedades da RdPCeLP

De acordo com [Rocha e Baranauskas 2003], é uma atitude irresponsável não efetuar alguma forma de avaliação em uma interface gráfica de um sistema computacional centrado no usuário, mesmo que certo grau de incerteza permaneça após testes exaustivos.

Para avaliar um sistema cuja interface foi modelada com RdPCeLP, serão verificadas se as propriedades da Rede de Petri estão presentes nas telas do sistema.

Dentre as formas de análise de comportamento de uma Rede de Petri, a RdPCeLP utiliza a simulação para efetuar a verificação das propriedades.

Porém, com a simulação as propriedades somente são avaliadas nos cenários que são simulados, não sendo, portanto completa. Com a simulação não é possível a previsão de todas as configurações possíveis do sistema, pois não é um método exaustivo, mas é possível testar o comportamento do sistema dado algumas configurações.

As propriedades inerentes às Redes de Petri, como alcançabilidade, vivacidade etc., podem ser testadas no modelo das RdPCeLP.

Entretanto, para que essas características sejam verificadas é necessário simulá-las.

Porém, como o conceito das Redes de Petri Coloridas com expressão da Lógica Proposicional adiciona condições e testes nos arcos, tal fato as difere, particularmente, das Redes de Petri Coloridas.

Por isso, não há, ainda, simuladores para a RdPCeLP.

Dessa forma, a simulação é feita apenas de forma manual, simulando o movimento das fichas coloridas através da rede e gerando assim diversos cenários possíveis.

Exemplo 15 A simulação desse sistema ocorre no disparo das sucessivas transições, gerando assim a cada disparo uma nova marcação. Todas as possibilidades de disparo e o movimento das fichas é uma simulação desse sistema. A simulação não é um método completo para verificar as propriedades, porém elas podem ser verificadas nos cenários simulados. As Figuras 4.17 e 4.18 demonstram uma RdPCeLP antes e depois do disparo da transição tr_1 .

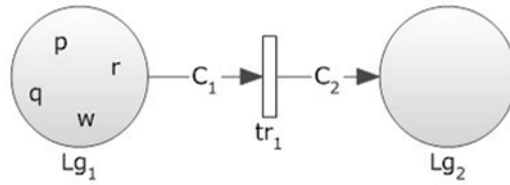


Figura 4.17: Simulação da rede, cenário antes do disparo de tr_1 .

Simulação da rede, cenário depois do disparo de tr_1

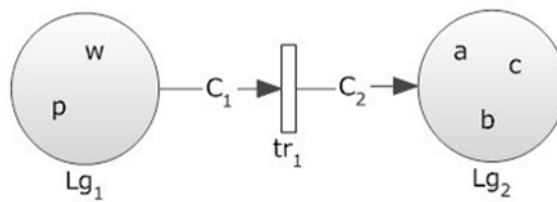


Figura 4.18: Simulação da rede, cenário depois do disparo de tr_1 .

4.9 Simulação em Rede de Petri das regras de ouro e heurísticas

Dentre as 8 regras de ouro de Shneiderman e as 10 heurísticas da usabilidade de Jakob Nielsen, algumas são representadas, a seguir, em uma RdPCeLP e simuladas para a verificação das propriedades da rede. Dentre as regras de ouro de Shneiderman pode-se destacar:

- **Permitir que os usuários possam usar atalhos** Como a interface deve prover aos usuários modos de acesso mais rápido, como uso de teclas especiais, é importante que na Rede de Petri sejam declaradas formas de passagem entre telas através do acionamento de teclas.

Esse tipo de requisito pode ser expresso na RdPCeLP.

Exemplo 16

Em um sistema acadêmico de educação à distância, por exemplo, podem-se prover teclas de atalho para acesso a recursos de um sistema.

Na Figura 4.19, o lugar Lg_1 contém teclas de atalho para acesso a recursos de um sistema.

Nesse caso, há duas fichas distintas para o acesso ao Fórum: as fichas coloridas *forum* e *atalho_forum*.

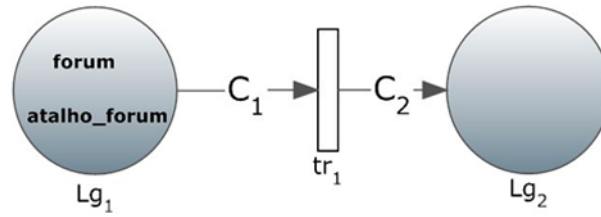


Figura 4.19: Atalho para recursos de Fórum.

Para o funcionamento do sistema com o uso dessas teclas de atalho, pode-se adicionar nas cláusulas o tratamento para *forum* e *atalho_forum*, da seguinte forma:

$$C_1 = forum \wedge atalho_forum$$

$$C_2 = forum \wedge topicos \wedge msgs$$

Essas cláusulas indicam que existe na tela representada por Lg_1 a opção de tecla de atalho.

Além disso, se ela for pressionada, a transição tr_1 dispara.

- **Oferecer feedback informativo**

Toda ação do usuário deve gerar algum *feedback*.

Isso significa que o usuário deve ser informado de todos os estados do sistema, inclusive proporcionar informações da posição no qual ele se encontra.

Exemplo 17

No caso da cláusula da Figura 4.19, pode-se simular a exibição do status do sistema, utilizando a cláusula a seguir:

$$C_2 = forum \wedge topicos \wedge msgs \wedge status_sist$$

A ficha colorida *status_sist* indica que na nova tela é exibida a informação de status do sistema.

A criação dessa nova ficha condiciona a nova tela a apresentar o status do sistema.

- **Oferecer prevenção e recuperação de erros**

O sistema deve evitar situações de erro e os casos nos quais se torna instável.

Para evitar esse tipo de problema é necessário que o sistema tenha um tratamento de erros, redirecionando, por exemplo, o usuário para uma página de erro com instruções, ou ajuda.

Exemplo 18

Considerando, por exemplo, uma página que efetua o envio de um formulário. Nesse envio pode ocorrer a geração de um erro e caso isso ocorra, tal fato deve ser tratado pelo sistema. Esse tratamento consiste na identificação do erro e a efetuação de uma operação adequada para esse erro. A Figura 4.20 ilustra um envio de formulário de contato.

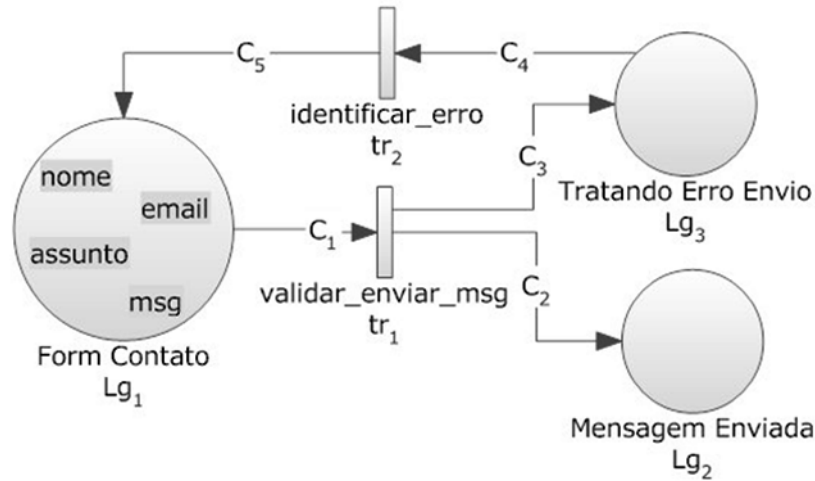


Figura 4.20: Rede de Petri para envio de formulário de contato.

Na Figura 4.20 o lugar Lg_1 é a tela do formulário de contato. Esse formulário contém os campos para o preenchimento de nome, e-mail, assunto e mensagem, que é identificado pelas fichas coloridas nome, email, assunto e msg respectivamente. A representação dessas funcionalidades é feita na rede pelas cláusulas:

$$C_1 = nome \wedge email \wedge assunto \wedge msg$$

$$C_2 = sucesso$$

$$C_3 = erro \wedge nome \wedge email \wedge assunto \wedge msg$$

$$C_4 = erro \wedge nome \wedge email \wedge assunto \wedge msg$$

$$C_5 = tipo_erro \wedge nome \wedge email \wedge assunto \wedge msg$$

A transição tr_1 executa a validação e envio da mensagem.

Esse evento pode gerar duas fichas coloridas novas: sucesso ou erro.

Caso ela gere sucesso a cláusula C_2 é executada alterando o estado do sistema para **Mensagem Enviada**, encaminhando uma ficha sucesso para o lugar Lg_2 .

Caso a transição tr_1 gere um erro, o estado do sistema é alterado para **Tratando Erro Envio**, encaminhando as informações fornecidas pelo usuário e uma mensagem de erro, representada por uma ficha erro no lugar Lg_3 .

- **Permitir desfazer facilmente as operações** As ações devem ser reversíveis.

Isso significa que um usuário pode querer retornar a um estado anterior caso deseje.

Exemplo 19

Considerando uma página que efetua o envio de um formulário.

Antes desse envio, o sistema pode solicitar que o usuário confirme o envio.

Caso o usuário não deseje enviar, o sistema deve identificar isso e retornar para um estado anterior.

A Figura 4.21 ilustra a confirmação de envio de uma mensagem de contato.

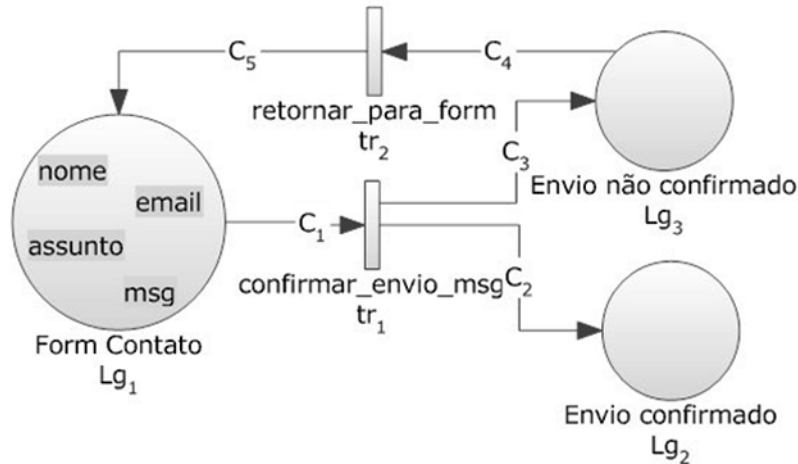


Figura 4.21: Rede de Petri para confirmação de envio de formulário de contato.

Na Figura 4.21, o lugar Lg_1 é a tela do formulário de contato.

Esse formulário contém os campos para o preenchimento de nome, email, assunto e mensagem, que é identificado pelas fichas coloridas *nome*, *e-mail*, *assunto* e *mensagem*, respectivamente.

A representação dessas funcionalidades é feita na rede pelas cláusulas:

$$C_1 = nome \wedge email \wedge assunto \wedge msg$$

$$C_2 = envio_confirmado \wedge nome \wedge email \wedge assunto \wedge msg$$

$$C_3 = envio_no_confirmado$$

$C_4 = \text{envio_no_confirmado}$

$C_5 = \text{nome} \wedge \text{email} \wedge \text{assunto} \wedge \text{msg}$

O disparo da transição tr_1 solicita ao usuário a confirmação do envio de uma mensagem.

Esse evento pode gerar duas fichas coloridas novas: *envio_confirmado* ou *envio_não_confirmado*.

Caso ela gere a ficha colorida *envio_confirmado* a cláusula C_2 é executada, alterando o estado do sistema para **Envio Confirmado**.

Nesse caso, são encaminhados os dados fornecidos pelo usuário e uma ficha *envio_confirmado* para o lugar Lg_2 .

Caso a transição tr_1 gere uma ficha colorida *envio_não_confirmado* o estado do sistema é alterado para **Envio não confirmado**.

Nesse caso, não é necessário encaminhar as informações fornecidas pelo usuário, pois a ação será cancelada.

Isso significa o prosseguimento da execução, no sistema, somente da ficha *envio_não_confirmado*.

Algumas das 10 heurísticas de Nielsen também podem ser representadas por uma Rede de Petri Colorida com expressões da Lógica Proposicional.

Essa representação é considerada a seguir.

- **Liberdade e controle do usuário**

Deve permitir ao usuário desfazer ou refazer ações. Poder voltar para um ponto onde estava, quando estiver perdido em situações inesperadas. Assim, o usuário pode escolher se deseja ou não prosseguir com uma ação.

Essa heurística é similar à regra de ouro de Shneiderman, **Permitir desfazer facilmente as operações**.

Na simulação de uma RdPCeLP pode-se verificar nos cenários gerados se a propriedade da reversibilidade está presente.

Essa propriedade identifica se a Rede de Petri é, ou não, reversível caso a marcação inicial volte a ocorrer ao longo das sequências de disparo.

Como na análise dos modelos de fluxo de navegação é desejável que o usuário possa retornar sempre para a tela inicial do sistema, então o conceito de reversibilidade deve ser satisfeito. Ou seja, é necessário garantir que qualquer marcação desejável possa voltar a ocorrer.

Exemplo 20 O exemplo apresentado na Figura 4.21 considera a regra de ouro de Shneiderman: “Permitir desfazer facilmente as operações”.

Ele apresenta uma rede que possibilita o retorno a um estado anterior. Permite, assim, que o usuário possa confirmar se uma operação é, ou não, realizada. Caso a operação não seja confirmada, o sistema retorna para um estado anterior.

- **Prevenção de erros**

Ações como exclusões de itens, ou solicitações devem sempre vir acompanhadas com diálogos para confirmar a certeza na execução de uma ação. Tal fato evita erros. Dessa forma, o usuário deve possuir meios para retornar a um estado anterior.

Essa heurística é similar às regras de ouro de Shneiderman: **Oferecer prevenção e recuperação de erros**; e **Permitir desfazer facilmente as operações**.

- **Ajuda e documentação**

Um bom design evita ao máximo a necessidade de ajuda na utilização do sistema.

Entretanto, sistemas de ajuda devem existir para orientar o usuário. Por isso, todas as telas do sistema devem possuir o recurso de ajuda.

Exemplo 21

Na validação do sistema, deve-se verificar todos os cenários gerados. E em todos os cenários, as telas ativadas devem possuir uma ficha colorida chamada, por exemplo, *ajuda*.

Esse tipo de verificação é feita por meio de simulação, na qual o máximo de cenários possíveis é testado.

Capítulo 5

Trabalhos Relacionados

Para a realização desta dissertação a revisão bibliográfica contou com diversos modelos que pretendem modelar a navegabilidade em interface de software interativo. A seguir são identificados alguns trabalhos relacionados a esta dissertação:

- **LOCPN** (*Learning Objects production with Colored Petri Nets*) [Souza et al. 2007]: apresenta uma abordagem para diagramação de fluxo de navegação baseado em Rede de Petri Colorida. Esse foi o artigo que influenciou na criação da Rede de Petri Colorida com expressões da Lógica Proposicional, que foi uma das propostas deste trabalho.
- **PDIWA** (*Processo de Design de Interface Web Adaptativa*) [Batista e Ulbricht 2010]: O PDIWA é um modelo e diretrizes para o design de sistemas que se adaptem às necessidades do usuário. Esse processo marca as etapas do desenvolvimento de um software, e dentre essas etapas a elicitação de requisitos e o uso dessa elicitação para a modelagem da navegabilidade de um sistema. Foi escolhido como um trabalho relacionado, pois essa dissertação também apresenta uma definição de etapas dentro de um projeto de desenvolvimento de um sistema, porém essa dissertação se limita nas etapas de levantamento de requisitos e modelagem de fluxo de navegação.
- **MoLIC** (*Modeling Language for Interaction as Conversation*) [Sangiorgi e Barbosa 2009]: A MoLIC é uma linguagem de modelagem de interação que usa a metáfora da conversação, como se o usuário fosse o interlocutor em uma conversa com o sistema. Utiliza diagramas para representar essa conversa, e é um trabalho relacionado por conta disso.
- **OOHDM** (*Object-Oriented Hypermedia Design Method*) [Schwabe et al. 1996]: O OOHDM é um modelo de navegação e abstração de interface, utilizando uma abordagem orientada a objetos. Compreende etapas de desenvolvimento, que são o design conceitual, o design da navegação, o design da interface abstrata e a implementação. Foi escolhido como um dos trabalhos relacionados, pois é a modelagem de navegação mais utilizada pelos projetistas de interface.

5.1 LOCPN (*Learning Objects production with Colored Petri Nets*)

A metodologia LOCPN [Souza et al. 2007] é um modelo formal desenvolvido para auxiliar o processo de desenvolvimento de objetos de aprendizagem, foi baseado na Rede de Petri Colorida (RPC). Utiliza a mesma notação da RPC, com lugares, fichas coloridas e transições.

No LOCPN alguns lugares representam telas / janelas de implementação de um Objeto de Aprendizagem, elas podem representar tanto telas quanto *pop-ups* e caixas de diálogo. Na Figura 5.1 são representadas duas telas, **Tela1** e **Tela2**.

As fichas representam controles (widgets) associados às telas, como por exemplo, botões. Na Figura 5.1 são representados 3 botões na **Tela1** e um botão na **Tela2**. O modelo possui uma ficha importante, chamada de ficha *e*, que indica onde está o foco da aplicação, na Figura 5.1 nota-se que o foco está na **Tela1**, pois ela contém a ficha *e*.

As transições atuam em lugares que representam telas, e devem ser marcadas com pesos, que indicam ações sobre os controles da tela ou mudança de foco, passando assim a ficha *e*. Na Figura 5.1 observa-se uma transição que representa a utilização do botão **btOK** da **Tela1**, e com isso a geração de uma ficha *e* que significa a mudança de foco da **Tela1** para a **Tela2** do modelo.

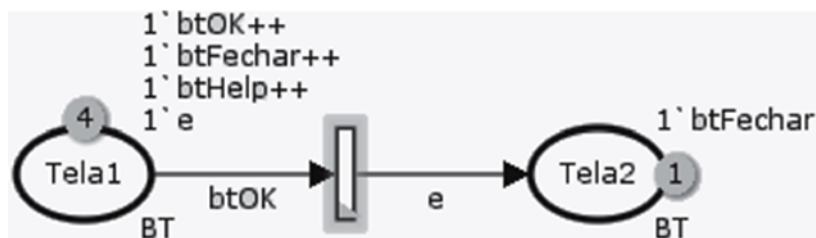


Figura 5.1: Representação de uma Transição entre telas [Souza et al. 2007].

A Figura 5.2 apresenta uma nova notação apresentada no trabalho de [Souza et al. 2007], no qual os retângulos com bordas duplas representam as telas do modelo, as conexões são representadas pelos lugares A e B, que [Souza et al. 2007] define como pontos de entrada e saída de informações entre modelagens individuais, e servem de “cola” entre as especificações.

A partir do trabalho que descreve o modelo LOCPN iniciaram-se as pesquisas para a definição do modelo RdPCeLP apresentado nesse trabalho. Verificou-se que o modelo estudado é de interesse na modelagem de fluxo de navegação, porém algumas estruturas possuem um uso redundante, como por exemplo, a elipse da Figura 5.1 possui a mesma função do retângulo com borda dupla da Figura 5.2. Algumas nomenclaturas nas figuras não ficaram claras, como por exemplo, o “++” que acompanha os botões da **Tela1** da Figura 5.1. O modelo quando aplicado em exemplos mais complexos formam diagramas complexos e difíceis de entender.

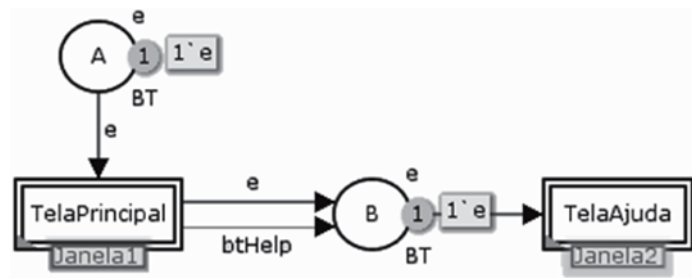


Figura 5.2: Representação Hierárquica da Estrutura de um Objeto de Aprendizagem [Souza et al. 2007].

O uso da Rede de Petri Colorida se mostrou eficiente no LOCPN, o que motivou sua extensão pelo modelo RdPCeLP. O uso de expressões lógicas ofereceu mais poder à modelagem, uma vez que se pode testar também quando algo está inativo, ou mesmo habilitar ou desabilitar elementos, poder que o LOCPN não tem como representar. O uso da ficha e é interessante, porém sobrecarrega o modelo, uma vez que essa ficha tem que ser repassada constantemente no modelo. O RdPCeLP não possui a representação do foco, algo que o LOCPN abrange, um modo de modelar o foco atual do uso do sistema é um dos trabalhos futuros sugeridos no Capítulo 6.

5.2 PDIWA (*Processo de Design de Interface Web Adaptativa*)

No trabalho de [Batista e Ulbricht 2010] é proposto um Processo de Design de Interface para a Web Adaptativa (PDIWA). Esse processo objetiva guiar o projetista ao longo de fases para o design de interfaces web adaptativas. O PDIWA apresenta a preocupação em oferecer modelos e um processo bem definido para auxiliar o projetista no desenvolvimento de interfaces, em especial interfaces adaptativas.

Uma hipermídia adaptativa promove uma melhor assistência à heterogeneidade de perfis de usuários. Assim o PDIWA apresenta um conjunto de tarefas que devem ser realizadas por uma equipe de desenvolvimento de interfaces para a elaboração de hipermídia adaptativa. Estas etapas são:

- **Análise:**

A etapa de Análise é o levantamento dos dados para construir os modelos do usuário, do domínio, da navegação e adaptação.

- **Conceito:**

Na etapa Conceito o projetista lista os requisitos da interface e define o estilo de navegação.

- **Desenvolvimento:**

A etapa de Desenvolvimento é a modelagem funcional, no qual a interface é criada.

- **Protótipo:**

Na etapa de Protótipo é feita a produção e tratamento das imagens, edição de vídeos, implementação dos modelos (templates) e armazenamento dos dados e mídias no banco de dados.

- **Teste:**

Na etapa de Teste são realizados os testes de usabilidade, da compreensibilidade dos ícones da interface e performance do sistema em diferentes navegadores.

Na Figura 5.3 o modelo PDIWA é apresentado, no qual é possível se obter uma visão geral do processo de design.



Figura 5.3: Modelo do Processo de Design de Interface web Adaptativa [Batista 2003].

O trabalho de [Batista e Ulbritch 2010] foi escolhido como um trabalho relacionado a esta dissertação, pois apresenta bem definidas as etapas do desenvolvimento de interface.

Nesta dissertação apresenta-se a árvore de características, que representa um **Modelo do Domínio** e também faz parte do **Modelo da Apresentação**, enquadrados nas etapas de Análise e Conceito, de acordo com o PDIWA (Figura 5.3). Pode ser considerado como parte do **Modelo do Domínio**, pois representa o conteúdo, as mídias que serão apresentadas e a organização da informação. Pode ser considerado como parte do **Modelo da Apresentação** principalmente por ser uma modelagem gráfica para o levantamento dos requisitos da interface.

A Rede de Petri Colorida com expressões da Lógica Proposicional representam o **Modelo de Navegação** (etapa Análise), pois possui estruturas de navegação e percursos possíveis. Representa também parte do **Modelo de Apresentação** (etapa Conceito), pois define o estilo de interação, representa os objetos de interação e a organização da apresentação.

5.3 MoLIC (*Modeling Language for Interaction as Conversation*)

A MoLIC é uma linguagem utilizada para modelar a interação do usuário com os sistemas computacionais [de Paula 2003].

No trabalho de [Sangiorgi e Barbosa 2009] é proposta uma extensão da MoLIC, que se destina a desenvolver modelos que representam a interface com o usuário. Ela é baseada na técnica de *storyboarding*, no qual a partir dos esquemas de telas é simulado o fluxo de navegação.

Na Figura 5.4 é apresentado um diagrama de interação básico em MoLIC, que representa a interação como uma conversa do usuário com o sistema. O MoLIC é um modelo semi-estruturado, utilizando a linguagem natural aliada a uma estrutura organizada e notações bem definidas.

Na Figura 5.4 os elementos diagramáticos são representações como segue [Sangiorgi e Barbosa 2009]:

- **Letra a** - Acesso ubíquo: uma oportunidade para o usuário modificar o assunto da conversa.
- **Letra b** - Expressão do usuário: identifica a ação do usuário em continuar em um tópico de conversa.
- **Letra c** - Cena: Uma cena é um momento de uma interação no qual o usuário decide como a conversa deve seguir. A primeira parte é o tópico da conversa e a segunda parte contém o diálogo.
- **Letra d** - Processos do sistema: A caixa preta identifica um processamento interno

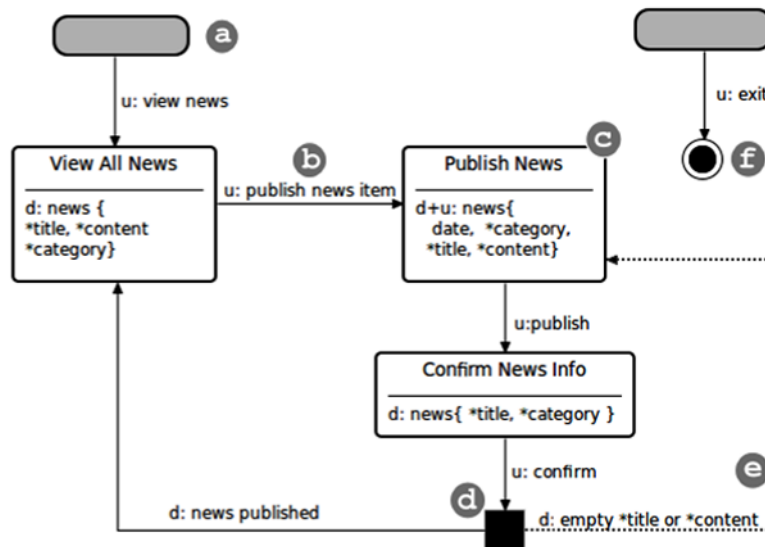


Figura 5.4: Um diagrama de diálogo básico em MoLIC [Sangiorgi e Barbosa 2009].

do sistema.

- **Letra e** - Expressão do projetista: É uma resposta do sistema a uma requisição do usuário.

- **Letra f** - Ponto de fechamento: representa o fim da conversa.

Os diagramas baseados em MoLIC e RdPCeLP são nitidamente distintos: o primeiro se baseia em um fluxo de interação, representado como uma conversa do usuário com o sistema, já o segundo é formal, baseando-se somente nos elementos presentes em uma interface e sua ativação. Porém, a RdPCeLP e seu tratamento direto com a manipulação dos elementos da interface é uma forma de conversa do usuário com o sistema com a automática resposta deste sistema e as mudanças na navegação.

Os modelos baseados em MoLIC somente podem ser validados a partir de simulações, pois não são modelos formais. Já no RdPCeLP, há o esforço pela formalização dos modelos com o uso de expressões da Lógica Proposicional e eficientes mecanismos obtidos a partir da Rede de Petri.

5.4 OOHDM (*Object-Oriented Hypermedia Design Method*)

O OOHDM foi proposto por Daniel Schwabe e Gustavo Rossi, da PUC-RJ, e descende da modelagem *Hypermedia Design Method* - HDM. Esse modelo propõe o processo de engenharia de desenvolvimento de sistemas em cinco etapas, sendo elas: Levantamento de Requisitos, Modelagem Conceitual, Modelagem Navegacional, Projeto de Interface Abstrata e Implementação.

A etapa a ser examinada e comparada com o RdPCeLP é a Modelagem Navegacional,

ela consiste na definição de como o usuário irá percorrer a interface de um sistema. De acordo com [Schwabe et al. 1996] o modelo utiliza tipos pré-definidos de classes navegacionais: nós, links e estruturas de acesso. A semântica de nós e links são os comumente utilizados em aplicações da Web, já as estruturas de acesso representam meios possíveis de acesso a nós.

A Figura 5.5 exibe um modelo baseado no OOHDM, nela temos um índice de notícia corrente que é dinâmico que é representado pelo retângulo preto na parte direita de índice, definindo que os elementos serão consultados em um banco de dados.

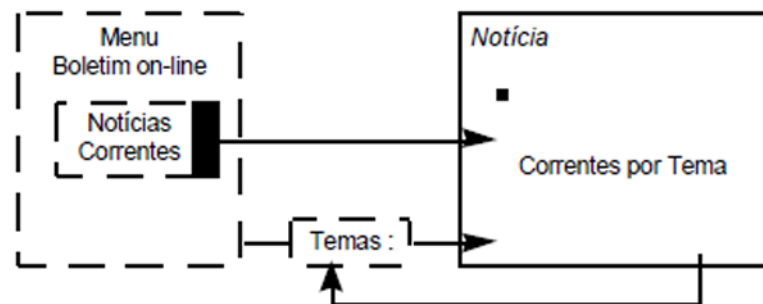


Figura 5.5: Exemplo de Modelagem OOHDM [Barroso e Schwabe 1998].

As Redes de Petri Coloridas com Expressões Lógicas diferem do modelo baseado no OOHDM pois possuem condições para a mudança de telas. O OOHDM se baseia em diversas representações gráficas para representar o funcionamento do sistema, enquanto que no RdPCeLP possui somente dois tipos de nós, os lugares, e transições. Assim, em uma primeira vista o modelo RdPCeLP dispense menos esforço no entendimento e pouco conhecimento da estrutura, porém o OOHDM demanda conhecimento do modelo e entendimento dos elementos constituintes para compreensão do funcionamento do sistema.

Outros trabalhos relacionados são apontados no Capítulo 2, na seção 2.4.1, em fundamentos de Interface Humano-Computador.

Capítulo 6

Conclusão

6.1 Síntese do Trabalho e Discussão

A modelagem de fluxo de navegação em sistemas computacionais centrados no usuário constitui um importante problema em Interface Humano Computador. Diversos trabalhos apresentam soluções para esse contexto.

Nesta dissertação, é introduzida uma proposta para modelagem de fluxo de navegação baseada em Rede de Petri Colorida, objetivando gerar uma técnica de modelagem que possa ser utilizada para esquematizar e testar interfaces de software.

Foi apresentada também uma técnica de modelagem de requisitos, denominada de árvore de característica, que possibilita a usuários de sistemas de computação a elicitação dos requisitos do sistema propriamente dito. Assim, como os usuários são os detentores do conhecimento no domínio do software, a modelagem proposta facilita a aquisição do conhecimento necessário ao desenvolvimento do sistema.

As árvores de características disponibilizam ao projetista uma técnica simples e prática para levantar os requisitos, sem confundir o cliente com definições técnicas.

A técnica de modelagem de requisitos proposta nesta dissertação foi aplicada a usuários com pouco conhecimento em informática, o que possibilitou averiguar que a mesma é de fácil assimilação. Isso, porque foi possível efetuar o desenvolvimento da árvore de características mesmo sem a assessoria aos usuários.

Um estudo foi realizado para avaliar a eficiência do modelo para a elicitação de requisitos. Nesse estudo participaram 44 professores da rede municipal de ensino da cidade de Uberlândia, esquematizando, cada um, uma árvore de característica, indicando assim que a técnica foi válida e o processo de desenvolvimento da árvore foi compreendido pelos participantes, demonstrando assim que o objetivo de ser uma ferramenta útil na elicitação de requisitos foi cumprido.

Foi demonstrado o uso da árvore de característica para a extração de requisitos de sistema que possam representar telas de sistemas, e tendo capturado esses requisitos,

esses foram traduzidos e relacionados em um grafo de telas, cujo objetivo é identificar a dinâmica do sistema, para assim conhecer como será o funcionamento geral da interface do futuro sistema computacional. A partir desse grafo de telas uma modelagem de fluxo de navegação baseada em Rede de Petri Colorida com expressões da Lógica Proposicional foi construída.

Com o grafo de telas pronto é possível aumentar seu nível de detalhamento e especificar o relacionamento entre as telas, indicando assim o fluxo de navegação entre elas. Assim, o grafo de telas é especificado na linguagem da Rede de Petri Colorida com expressões Lógicas, para assim haver uma melhor definição dos recursos de telas e as condições do fluxo dentro do sistema.

Este trabalho utilizou a Rede de Petri Colorida com expressões Lógicas e suas propriedades herdadas tanto da Rede de Petri Ordinária quanto da Lógica Proposicional, para testar propriedades desejáveis em interfaces, validadas pelas heurísticas de Nielsen e as regras de ouro de Shneiderman.

6.2 Trabalhos Futuros

Com a pesquisa e desenvolvimento desta dissertação foi possível vislumbrar diversas possibilidades de otimização e extensões que podem gerar importantes pesquisas futuras. Algumas dessas extensões e otimizações são apontadas a seguir:

- Desenvolvimento de uma formalização matemática para a árvore de características. Dessa forma, a modelagem utilizando árvores de características poderá alimentar a Rede de Petri de forma automática;
- Implementação de um sistema de software que auxilie a utilização das árvores de características por usuários leigos em computação. Assim fornecendo um meio gráfico para a criação, edição e impressão das árvores de características, automatizando o processo de desenvolvimento;
- Expandir o modelo da Rede de Petri Colorida com expressões da Lógica Proposicional a fim de que seja identificado de forma explícita em qual tela o usuário está posicionado. Nesse caso, a rede identificará, explicitamente, a tela ativa, ou que possui o foco da atividade.
- Desenvolver um simulador de Rede de Petri Colorida com expressões da Lógica Proposicional, no qual o usuário poderá gerar telas de sistema com auxílio da ferramenta, e a partir da geração dos esquemas de telas o sistema irá gerar automaticamente a Rede de Petri Colorida com expressões da Lógica Proposicional. A partir da geração dessa rede o sistema irá simular e testar propriedades importantes para o correto funcionamento do sistema, sugerindo assim melhorias e indicando erros.

- Simular a navegação em telas, utilizando o simulador de Redes de Petri Coloridas com expressões Lógicas.

Uma importante melhoria para a linguagem das Redes de Petri Coloridas com expressões Lógicas e a adição de estruturas condicionais nas transições. Uma transição dentro do modelo executa eventos e tem o "poder" de gerar fichas coloridas. Porém se, ao executar um evento, ocorrer um erro, a transição deverá poder gerar uma ficha de erro. No modelo atual pode ocorrer um problema, como ilustrado pela Figura 6.1.

Exemplo de geração de erro

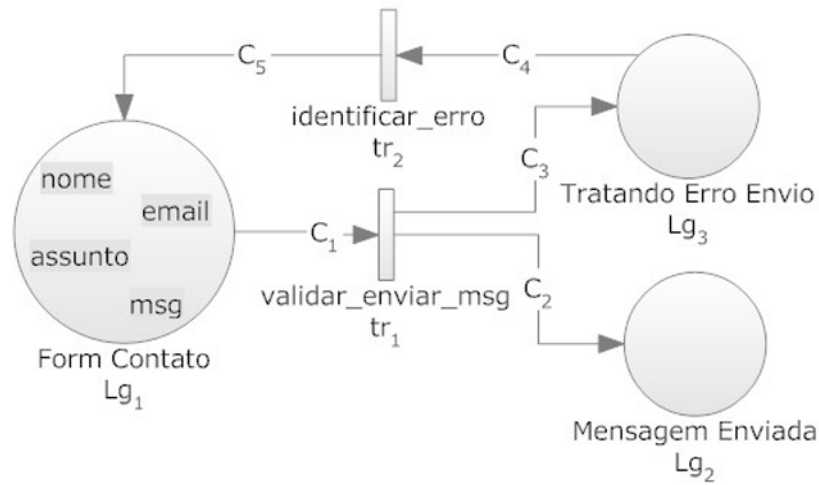


Figura 6.1: Exemplo de geração de erro.

As cláusulas da rede representada na Figura 6.1 são dadas por:

$$C_1 = nome \wedge email \wedge assunto \wedge msg$$

$$C_2 = sucesso$$

$$C_3 = erro \wedge nome \wedge email \wedge assunto \wedge msg$$

$$C_4 = erro \wedge nome \wedge email \wedge assunto \wedge msg$$

$$C_5 = tipo_erro \wedge nome \wedge email \wedge assunto \wedge msg$$

Percebe-se que tr_1 é executada, pois: $I[C_1, Lg_1] = T$. Porém ao disparar, a transição tr_1 executará C_2 e C_3 , gerando simultaneamente as fichas **sucesso** e **erro**, porém ela deveria gerar **sucesso** ou **erro**. Para isso deverá haver nessa transição uma forma de escolha, no qual **sucesso** ou **erro** fossem geradas, e arcos fossem encaminhados para lugares distintos. A solução para esse problema é uma sugestão para um trabalho futuro.

Referências Bibliográficas

- [Alves 2006] Alves, S. V. L. (2006). Suporte à Percepção em Groupware Síncronos de Aprendizagem. Master's thesis, Centro de Informática. Universidade Federal de Pernambuco, Recife, PE.
- [Azevedo et al. 2000] Azevedo, P., Merrick, R., e Roberts, D. (2000). OVID to AUIML - user-oriented interface modeling. In *TUPIS'2000 Workshop at the UML*.
- [Barroso e Schwabe 1998] Barroso, N. e Schwabe, D. (1998). Projeto de Navegação em Aplicativos Hipermídia Orientado ao Usuário. Disponível em www-di.inf.puc-rio.br/~schwabe/papers/MCC-10-98.pdf.gz.
- [Batista 2003] Batista, E. A. (2003). Uma Taxonomia Facetada para Técnicas de Elicitação de Requisitos. Master's thesis, UNICAMP.
- [Batista e Ulbricht 2010] Batista, R. e Ulbricht, V. R. (2010). Modelo e Diretrizes para o processo de design de interface web adaptativa. *10º USIHC- Análise do 10º Congresso Internacional de Ergonomia e Usabilidade de Interfaces Humano- Computadores*.
- [Booch et al. 2000] Booch, G., Rumbaugh, J., e Jacobson, I. (2000). *UML: guia do usuário*. Campus. Tradução por Fábio Freitas.
- [Campos 2004] Campos, J. (2004). The modelling Gap Between Software Engineering and Human-Computer Interaction. In Rick Kazman, L. B. e John, B. (editores), *ICSE 2004 Workshop: Bridging the Gaps II: Bridging the Gaps between Software Engineering and Human-Computer Interaction*, pp. 54-61, Edinburgh, Scotland.
- [Cardoso e Valette 1998] Cardoso, J. e Valette, R. (1998). *Redes de Petri*. Editora da UFSC, Santa Catarina.
- [Coppin 2010] Coppin, B. (2010). *Inteligência Artificial*. LTC, Rio de Janeiro.
- [Cordeiro et al. 2009] Cordeiro, R., Oliveira, M. R., e Chanquini, T. (2009). *Utilização de conceitos de interface homem-máquina para adaptação da disciplina de requisitos do RUP*. São Paulo.
- [De Bortoli e Price 2000] De Bortoli, L. e Price, A. M. A. (2000). O uso de Workflow para Apoiar a Elicitação de Requisitos.
- [de Paula 2003] de Paula, M. G. (2003). Projeto de Interação Humano-Computador Baseado em Modelos Fundamentados na Engenharia Semiótica. Master's thesis, Departamento de Informática/ PUC-Rio.

- [de Souza et al. 1999] de Souza, C. S., Leite, J. C., Prates, R. O., e Barbosa, S. D. J. (1999). Projeto de Interfaces do Usuário: perspectivas Cognitivas e Semióticas. In *Anais da Jornada de Atualização em Informática, XIX Congresso da Sociedade Brasileira de Computação*, Rio de Janeiro.
- [Dix 2004] Dix, A. e. a. (2004). *Human-computer interaction*. 3.ed edition.
- [Filho et al. 2004] Filho, I. M., Olliveira, T., e Lucena, C. J. P. d. (2004). A framework instantiation approach based on the Features Model. *the Journal of Systems and Software*.
- [Finkelstein et al. 1992] Finkelstein, L., Huang, J., Inkelstein, A., e Nuseibeh, B. (1992). Using Software Specification Methods for Measurement Instrument Systems. *Part 1: Structured Methods*. London: City University.
- [Fisher 2001] Fisher, M. C. B. O. (2001). Estudos de Requisitos para um Software Educativo de Apoio ao Ensino da Introdução à Computação:. Master's thesis, Universidade de São Paulo (USP).
- [Harel 1987] Harel, D. (1987). Statecharts: A Visual Approach to Complex Systems. *Science of Computer Programming*, 8:231–274.
- [Harel et al. 2000] Harel, D., Kozen, D., e Tiuryn, J. (2000). *Dynamic Logic*.
- [Hartson et al. 1990] Hartson, H., Siochi, A., e Hix, D. (1990). The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs. *ACM Transaction on Information Systems*, 8(3):181–203.
- [Hix e Hartson. 1993] Hix, D. e Hartson., H. R. (1993). *Developing User Interfaces: Ensuring Usability Through Product & Process*. New York, USA.
- [Huth e Ryan 2008] Huth, M. e Ryan, M. (2008). *Lógica em Ciência da Computação: modelagem e argumentação sobre sistemas*. Rio de Janeiro, 2ª edição edition. tradução e revisão técnica Valéria de Magalhães Iório.
- [Jensen 1992] Jensen, K. (1992). Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. *EARCS Monographs on Theoretical Computer Science*, 1. Springer-Verlag, Berlin.
- [Kieras 1999] Kieras, D. (1999). A Guide to GOMS Model Usability Evaluation using GOMSL and GLEAN3. Disponível em: <ftp.eecs.umich.edu/people/kieras>.
- [Kristensen et al. 1998] Kristensen, L. M., Christensen, S., e Kurt, J. (1998). The practitioners guide to coloured Petri Nets.
- [Limbourg et al. 2004] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., e Trevisan, M. F. D. (2004). Usixml: a user interface description language for context-sensitive user interfaces. In In Luyten, K, A. M. L. Q. e Vanderdonckt, J. (editores), *Proceedings of the ACM AVI2004 Workshop Developing User Interfaces with XML: Advances on User Interface Description Languages*, pp. 55–62, Gallipoli.
- [Machado 2006] Machado, T. (2006). Visual IMML: um perfil UML para modelagem de interfaces de usuário. Master's thesis, Universidade Federal do Rio Grande do Norte. Natal, RN.

- [Moran 1981] Moran, T. P. (1981). The command language grammar: a representation for the user interface of interactive computer systems. *Internacional Journal of Man-Machine Studies*.
- [Murata 1989] Murata, T. (1989). Petri Nets: Properties, Analysis and applications. *IEEE*, 77(4):541–580.
- [Neto et al. 2003] Neto, G. d. C., Gomes, A. S., e Tedesco, P. (2003). Elicitação de Requisitos de Sistemas Colaborativos de Aprendizagem Centrada na Atividade de Grupo. *SBIE 2003*. Disponível em acessado <<http://br-ie.org/pub/index.php/sbie/article/view/262/0>> Acesso em 23/02/2010.
- [Nielsen 1993] Nielsen, J. (1993). *Usability Engineering*. Boston.
- [Oliveira et al. 2010a] Oliveira, C. C., Oliveira, D. C., Oliveira, C. F., Cattelan, R., e Souza, J. N. d. (2010a). Comunidade Virtual de Aprendizagem para Crianças. In *IX Workshop de Ferramentas e Aplicações WEBMEDIA*, Belo Horizonte, MG.
- [Oliveira et al. 2010b] Oliveira, C. C., Oliveira, D. C., Oliveira, C. F., Cattelan, R., e Souza, J. N. d. (2010b). *Desenvolvimento de Gerenciador de Conteúdo com Tecnologia Web 2.0*. In: *Tópicos em Sistemas Colaborativos, Interativos, Multimídia, Web e Banco de Dados*, pp. 237–276. Belo Horizonte.
- [Oliveira et al. 2010c] Oliveira, C. C., Oliveira, D. C., Oliveira, C. F., Cattelan, R., e Souza, J. N. d. (2010c). Histórias que ensinam: narrando a teoria. In *III Encontro Mineiro de Psicologia Escolar e Educacional*, Uberlândia, MG.
- [Oliveira et al. 2010d] Oliveira, C. C., Oliveira, D. C., Oliveira, C. F., Cattelan, R., e Souza, J. N. d. (2010d). Árvore de Características de Software Educativo: Uma Proposta para Elicitação de Requisitos pelo Usuário. In *XXI Simpósio Brasileiro de Informática na Educação*, João Pessoa, Pb.
- [Osborn 1962] Osborn, A. F. (1962). *O Poder Criador da Mente: Princípios e processos do Pensamento Criador e do Brainstorming*. IBRASA, São Paulo.
- [Pais 2004] Pais, R. M. C. (2004). Geração de Executores e Analisadores de Redes de Petri. Master's thesis, Departamento de Informática/ Faculdade de Ciências e Tecnologia/Universidade Nova de Lisboa.
- [Parnas 1969] Parnas, D. L. (1969). On the Use of Transitions Diagrams in the Design of a User Interface for an Interactive Computer System. In *24th National ACM Conference*, pp. 379–385.
- [Paternò 2001] Paternò, F. (2001). Towards a uml for interactive systems. In *EHCI01: Proceedings of the 8th International Conference on Engineering for Human-Computer Interaction*.
- [Payne e Green 1989] Payne, S. e Green, T. (1989). *Task Analysis for Human-Computer Interaction*, chapter Task-Action Grammar: the Model and its Developments.
- [Petri 1966] Petri, C. A. (1966). Communication with Automata. *New York: Griffiss Air Force Base. Tech. Rep. RADC-TR-65-377*, 1.

- [Phanouriou 2000] Phanouriou, C. (2000). UIML: A Device-Independent User Interface Markup Language. Master's thesis, Virginia Polytechnic Institute and State University.
- [Preece et al. 2002] Preece, J., Rogers, Y., e Sharp, E. (2002). *Interaction Design: Beyond Human-computer Interaction*. John Wiley & Sons, New York, NY, 1st edition edition.
- [Pressman 2006] Pressman, R. (2006). *Engenharia de Software*. 6^a ed. edition.
- [Ramos e Oliveira 2009] Ramos, E. S. e Oliveira, J. M. P. (2009). Especificação e Verificação Formal de um modelo de STI-PBL por Redes de Petri Coloridas. In: *Revista Brasileira de Informática na Educação*, 17(3):53–66.
- [Rieder 2007] Rieder, R. (2007). Uma metodologia para Especificar Interação 3D utilizando Redes de Petri. Disponível em <http://grv.inf.pucrs.br/projects/PetriNets/DissertacaoRafaelFINAL.pdf>.
- [Rocha e Baranauskas 2003] Rocha, H. V. d. e Baranauskas, M. C. C. (2003). *Design e Avaliação de Interfaces Humano-Computador*. Campinas. NIED/UNICAMP.
- [Sangiorgi e Barbosa 2009] Sangiorgi, U. B. e Barbosa, S. D. (2009). MoLIC designer: towards computational support to hci design with MoLIC ACM. In *1st ACM SIGCHI Symposium on Engineering interactive Computing Systems. EICS' 09*, pp. 303–308, New York. DOI= <http://doi.acm.org/10.1145/1570433.1570489>.
- [Schwabe et al. 1996] Schwabe, D., Rossi, G., e Barbosa, S. D. J. (1996). Systematic Hypermedia Application Design with OOHDM. In Press, A. (editor), *Hypertext 96 Proceedings*, pp. 116–128, New York.
- [Shneiderman 1998] Shneiderman, B. (1998). *Designing the User Interface: Strategies for effective human-computer interaction*. Addison-Wesley.
- [Silva e Paton 2000] Silva, P. P. d. e Paton, N. W. (2000). UMLi: The Unified Modeling Language for Interactive Applications. In Proceedings «UML» 2000. *LNCS, Springer-Verlag*, 939:117–132.
- [Silva e Paton 2003] Silva, P. P. d. e Paton, N. W. (2003). User Interface Modeling with UML. *Department of Computer Science, University of Manchester*.
- [Silva 2007] Silva, S. F. d. S. (2007). BRIDGE: uma Ferramenta para o Design de Interfaces de Usuário baseada em Especificações IMML. Master's thesis, Departamento de Informática e Matemática Aplicada/UFRN.
- [Silva et al. 2005] Silva, S. M. A. d., Bonin, M. R., e Paludo, M. A. (2005). Levantamento de Requisitos Segundo o Método Volere. Disponível em <<http://publica.fesppr.br/index.php/rnti/article/viewFile/v1n1ART2/86>> Acesso em: 01/07/2009.
- [Silva e Silveira 2010] Silva, T. S. d. e Silveira, M. S. (2010). Validação de um Método para Identificação de Problemas de Usabilidade a partir de Diagramas UML. In *IHC 2010: IX Simpósio sobre Fatores Humanos em Sistemas Computacionais*, Belo Horizonte.
- [Silveira 2002] Silveira, M. S. (2002). *Processo de Design de IHC*. Faculdade de Informática/FACIN/PUCRS. Notas de Aula.

- [Sommerville 2007] Sommerville, I. (2007). *Engenharia de Software*. São Paulo, 8ª ed. edition.
- [Souza 2008] Souza, J. N. (2008). *Lógica para Ciência da Computação*. Campus, Rio de Janeiro, 2ª ed. edition.
- [Souza 2009] Souza, J. N. d. (2009). *Belisca no Mundo da Lógica*. Publicação interna da Universidade Federal de Uberlândia.
- [Souza et al. 2007] Souza, M. d. F. C. d., Gomes, D. G., Barroso, G. C., Souza, C. T. d., Filho, J. A. d. C., e Pequeno, M. C., A. R. (2007). LOCPN: Redes de Petri Coloridas na Produção de Objetos de Aprendizagem. *Revista Brasileira de Informática na Educação*, 15:39–52.