

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO



**MNLAC: UMA PROPOSTA DE MODELAGEM DE FLUXO
DE NAVEGAÇÃO BASEADA EM LÓGICA MODAL**

DANIELE CARVALHO OLIVEIRA

Uberlândia - Minas Gerais

2011

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO



DANIELE CARVALHO OLIVEIRA

MNLAC: UMA PROPOSTA DE MODELAGEM DE FLUXO DE NAVEGAÇÃO BASEADA EM LÓGICA MODAL

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
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada “**MNLAC: UMA PROPOSTA DE MODELAGEM DE FLUXO DE NAVEGAÇÃO BASEADA EM LÓGICA MODAL**” por **Daniele Carvalho Oliveira** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 25 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
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Data: Agosto de 2011

Autor: **Daniele Carvalho Oliveira**
Título: **MNLAC: UMA PROPOSTA DE MODELAGEM DE FLUXO
DE NAVEGAÇÃO BASEADA EM LÓGICA MODAL**
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.

Dedicatória

À minha família:

Meu pai Clério, minha madrastra Joana e minhas irmãs Cintia e Elisabete

Agradecimentos

Agradeço...

À Deus por todas as oportunidades que me são dadas. Por me dar o dom do amor e da vida.

À minha família o apoio, amor e compreensão.

Ao meu pai pelo incentivo e compreensão em cada momento da minha vida.

À minha madrastra Joana por todo amor e apoio que tem nos dado.

À minha irmã Cintia por ter me impulsionado e incentivado a concluir mais esta etapa.

À minha irmã Elisabete pelo carinho, amor e paciência.

À todos os professores da Faculdade de Computação que me transmitiram suas experiências e conhecimento.

Ao professor Renan pela orientação.

Ao professor João Nunes pelo apoio, amizade e orientação em todas etapas da realização deste trabalho.

*“Escolha um trabalho que você ame
e não terá que trabalhar um único dia em sua vida.”*

Confúcio

Resumo

Atualmente, diversas técnicas estão sendo exploradas com objetivo de melhorar o processo de projeto e desenvolvimento de interfaces com usuário. Novas técnicas criadas especificamente para interface humano-computador ou técnicas adaptadas da engenharia de software procuram suprir a deficiência na área. A complexidade dos requisitos exigidos pelo usuário e a dificuldade em entendê-los são as principais causas dos problemas enfrentados ao projetar a interface de um sistema.

Para projetar um bom sistema, não basta apenas focar em suas funcionalidades e em como elas serão implementadas, é essencial projetar, também, uma boa interface. Construir uma interface usável é um processo que demanda estudo e planejamento.

Este trabalho apresenta a formalização de um método de modelagem de interfaces com usuário que tem como objetivo simplificar o projeto de interfaces e verificar a satisfação de propriedades importantes e desejáveis do sistema. A especificação do método proposto neste trabalho, denominado MNLAC, torna possível definir formalmente, com Lógica Modal, a modelagem navegacional de um sistema com interface com usuário. O MNLAC é um método que permite a verificação de propriedades, utilizando as propriedades da Lógica Temporal, no fluxo de telas, possibilitando a identificação de erros como telas inalcançáveis, problemas de usabilidade e inconsistências.

Palavras chave: interface humano-computador, modelagem navegacional, fluxo de navegação, lógica modal, engenharia web

Abstract

Currently several techniques are being explored in order to improve the design process and development of user interfaces. New techniques developed specifically for human-computer interface or adapted techniques of software engineering try to remedy the shortcoming in the area. The complexity of user requirements and the difficulty in drawing them are the main causes of the problems faced when designing an interface system.

To design a good system, not enough focus on their features, and how they are implemented, it is essential to design also a good interface. Building a usable interface is a process that requires study and planning.

This paper presents the formalization of a user interface modeling method, which aims to simplify the interface design and verify the satisfaction of important and desirable system properties. The specification of the method proposed in this paper, called MNLAC, makes possible to formally define, with Modal Logic, the navigational modeling of a system with a user interface. The MNLAC is a method that allows the verification of properties using the properties of temporal logic, in the flow of screens, allowing the identification of errors such as unreachable screens, usability problems, and inconsistencies, among others.

Keywords:

human-computer interface, navigational modeling, navigation flow, modal logic, web engineering

Sumário

Lista de Figuras	xix
1 Introdução	21
1.1 Contexto	21
1.2 Motivação e Problema	22
1.3 Objetivos	23
1.4 Organização do Trabalho	24
2 Fundamentos	27
2.1 Fundamentos de lógica modal	27
2.1.1 Lógica Modal	27
2.1.2 Lógica Temporal Linear	39
2.1.3 Lógica de Árvore Computacional	45
2.2 Métodos de Modelagem de Interface	46
2.2.1 Métodos de Especificação de Interface	47
2.2.2 Métodos de Prototipação de Interface	52
3 MNLAC: Modelagem Navegacional Baseada em Lógica Modal	55
3.1 Elicitação de Requisitos	55
3.2 Geração do Grafo de Telas	56
3.3 Protótipo da Interface	58
3.4 Modelo Navegacional baseado em Lógica de Árvore Computacional	67
4 Trabalhos Relacionados	87
4.1 Object Oriented Hypermedia Design Method	87
4.1.1 Modelagem Navegacional	88
4.2 Modeling Language for Interaction as Conversation	90
4.3 OOHDM, MoLIC e a MNLAC	91
5 Conclusões e Trabalhos Futuros	93
5.1 Contribuições	93
5.2 Trabalhos Futuros	94

Referências Bibliográficas

97

Lista de Figuras

2.1	Frame	30
2.2	Modelo	33
2.3	Operadores $\diamond$ e $\square$	35
2.4	Transitividade do Operador $\diamond$	36
2.5	Transitividade do Operador $\square\diamond$	37
2.6	Transitividade do Operador $\diamond\square$	38
2.7	Exemplo do operador X	42
2.8	Exemplo do operador G	42
2.9	Exemplo do operador F	43
2.10	Exemplo do operador U	44
2.11	Exemplo do operador R	45
3.1	Árvore de características de um portal educativo	56
3.2	Grafo de Telas	58
3.3	Tela de Chat	61
3.4	Tela Principal	62
3.5	Tela Principal	62
3.6	Tela Chat e Fórum	66
3.7	Modelo Navegacional	68
3.8	Árvore de características modificada	84
3.9	Grafo de Telas modificado	84
4.1	Modelo Proposto pelo OOHDM	88
4.2	Exemplo de Nó	89
4.3	Exemplo de Elo	89
4.4	Exemplo de um esquema navegacional simples	90
4.5	Diagrama de diálogo básico em MoLIC	91
4.6	Exemplo de modelagem OOHDM	92

Capítulo 1

Introdução

Este trabalho propõe formalizar um método de modelagem de interfaces com usuário, com objetivo de simplificar o projeto de interfaces, focando na modelagem navegacional da interface de sistemas computacionais. É apresentado um método de modelagem que engloba desde o levantamento de requisitos, até o teste do fluxo navegacional utilizando Lógica Modal.

1.1 Contexto

Projetar um sistema computacional é mais que simplesmente criar modelos que representem funcionalidades e estados. É necessário principalmente ser capaz de averiguar a correção dos modelos, a fim de reduzir a ocorrência de erros de projeto que poderiam culminar em novo redesenho, acarretando mais gastos de tempo e recursos. Apesar dos métodos de Engenharia de Software serem aplicados cuidadosamente, com rigorosas revisões, e produzirem sistemas de alta qualidade, quaisquer descuidos na aplicação desses métodos podem acarretar em diversos problemas, como, por exemplo, contradições, ambiguidades, pontos vagos, declarações incompletas e níveis de abstrações misturados [Pressman 2006].

Métodos formais de modelagem fornecem modelos de análise mais completos, consistentes e não ambíguos do que aqueles produzidos usando métodos convencionais [Pressman 2006].

“Especificações formais podem ser estudadas matematicamente enquanto especificações informais não podem. Por exemplo, pode-se provar que um programa correto satisfaz suas especificações, ou pode-se provar que dois conjuntos alternativos de especificações são equivalentes, assim certas formas de incompleteza ou inconsistência podem ser detectadas automaticamente.” [Liskov e Berzins 1986]

Sabe-se que a matemática fornece um alto nível de validação, permitindo demonstrar

que um projeto satisfaz uma especificação, e que um código de programa é um reflexo correto de um projeto. [Pressman 2006]

“Os métodos formais tem um potencial tremendo para melhorar a clareza e a precisão da especificação dos requisitos e de encontrar erros importantes e sutis.” [Easterbrook et al. 1997]

1.2 Motivação e Problema

Para projetar um bom sistema, não basta apenas focar em suas funcionalidades, e em como será implementá-las, é importante projetar, também, uma boa interface. Diversos métodos estão sendo criados para auxiliar o processo de desenvolvimento de interface. Um desses métodos é o *Object-Oriented Hypermedia Design Method* [Schwabe e Rossi 1995] que é composto de quatro atividades focadas na representação dos objetos de uma interface com usuário e a estrutura navegacional da aplicação.

No entanto projetar uma boa interface com o usuário, fácil de usar, é um processo complexo. Embora os benefícios da melhora da usabilidade de uma interface sejam indiscutíveis, não são resolvidos os problemas que levam à dificuldade do projeto de uma Interface [Myers 1993]. Algumas dessas dificuldades são:

- Dificuldade em entender as tarefas e os usuários
Entender o que o usuário final quer e precisa é difícil, sobretudo porque a maioria dos programadores tem dificuldade de se imaginar na condição de usuários comuns.
- Complexidade inerente às tarefas e aplicações
O domínio da aplicação costuma ser amplo e de difícil modelagem, seja porque a tarefa de modelar é complicada, seja porque a aplicação se propõe a resolver muitos problemas.
- Variedade de aspectos e requisitos diferentes
Interfaces com o usuário envolvem questões como design gráfico, documentação, internacionalização e desempenho, o que contribui para aumentar a complexidade do desenvolvimento.
- Teoria e métodos não são suficientes para resolver o problema
A habilidade dos projetistas é fator primário para a qualidade das interfaces geradas. É difícil criar um método abrangente, grande parte dos projetos apresentam situações que são exceções às regras propostas nos modelos.
- Dificuldade de se fazer um projeto iterativo
O ideal é refinar uma interface com usuário ciclicamente, mas muitas vezes as modificações trazem uma piora de usabilidade, além da complexidade de obter resultados dos usuários finais em cada iteração. [Myers 1993]

Enfim, construir uma interface usável, ou seja, uma interface simples de usar e aprender, é um processo que demanda planejamento. Um dos problemas mais comuns no desenvolvimento de interfaces são problemas relacionados com fluxo de navegação, como por exemplo, links quebrados, operações que não podem ser executadas. Dessa forma, planejar a navegabilidade do sistema é essencial.

1.3 Objetivos

A proposta desenvolvida utiliza o formalismo da Lógica Modal para modelar interfaces com usuário. Assim, objetiva-se definir matematicamente o que compõe a interface com usuário de um sistema, de forma a possibilitar a verificação de propriedades necessárias a um bom funcionamento do software.

A Lógica, tipicamente, consiste de três ingredientes, [Harel et al. 2000]:

- **Linguagem**

Uma linguagem é uma coleção de expressões bem formadas, às quais se pode atribuir um significado. Os símbolos de uma linguagem, junto com regras formais para distinguir expressões de um agregado arbitrário de símbolos, são chamados de sintaxe da linguagem.

- **Semântica**

A semântica diz como interpretar expressões bem formadas como declarações sobre algo. Esse “algo” pode ser um objeto matemático como grupos, gráficos ou números naturais, ou elementos do cotidiano como carros, empregados, ou o tempo. Declarações de uma linguagem expressa as propriedades e relações entre esses objetos.

- **Sistema de dedução**

Um sistema de dedução consiste de uma coleção de regras que podem ser aplicadas para derivar, de modo mecânico, fatos interessantes sobre objetos semânticos e suas relações. Os fatos e relações são aqueles que são exprimíveis na linguagem.

Esses três aspectos, linguagem, semântica e sistema de dedução, podem ser adaptados para uma aplicação em particular e para determinados níveis de expressividade.

No caso da proposta ora apresentada, esses três elementos são adaptados para o estudo da navegabilidade de telas em sistemas de interface.

De forma análoga, em Engenharia de Software de modo geral, para realizar a validação formal de sistemas, são também necessários três requisitos [Huth e Ryan 2008]:

- Um *framework* para modelar sistemas, geralmente uma linguagem descritiva;
- Uma *linguagem de especificação* para descrever as propriedades a serem verificadas;

- Um *método de verificação* para verificar se a descrição de um sistema satisfaz a especificação.

Observa-se, então, que os três quesitos na Engenharia de Software consideram, como na Lógica, a linguagem, a especificação semântica e o método de verificação ou dedução.

Este trabalho também segue tais requisitos na apresentação do sistema proposto. Nesse caso, o modelo navegacional do fluxo de telas é descrito por uma linguagem, que possui uma semântica e tudo isso é associado a um sistema de dedução.

Sendo assim, a linguagem possibilita a representação das telas de uma forma simples. E tal linguagem também possibilita a análise do fluxo das telas ao longo da utilização do sistema de interface pelos usuários.

Nesse contexto, o modelo navegacional de telas é representado como um grafo de telas, no qual são especificadas as funções de cada tela. Como será visto mais adiante, uma tela representa a situação, estado, ou mundo possível da interface do sistema em determinado momento, sendo representada como um vértice do grafo de telas.

Além do grafo de telas a proposta apresentada utiliza a linguagem e os fundamentos da Lógica Modal. A Lógica Modal é utilizada como linguagem descritiva e de especificação do modelo proposto. Os fundamentos da dedução em Lógica Modal são utilizados para analisar o fluxo navegacional das telas, ou seja, para verificar propriedades do sistema.

Como caso de estudo, este trabalho apresenta a modelagem de um sistema educativo para o Ensino Fundamental [Oliveira et al. 2010a]. O sistema é uma comunidade virtual de aprendizagem voltada para o ensino de crianças, inicialmente projetado para o ensino de lógica para alunos do sexto ao nono ano do Ensino Fundamental.

Para o projeto deste portal foram utilizadas as teorias de aprendizagem de Vygotsky, que utiliza como base o ensino teórico através de histórias, a vivência dos conceitos adquiridos e estímulo de atitudes proativas para o desenvolvimento do raciocínio [Oliveira et al. 2010c]. O Portal está sendo desenvolvido na forma de um Gerenciador de Conteúdo para Web utilizando a Arquitetura MVC e tecnologia Web 2.0 [Oliveira et al. 2010b].

1.4 Organização do Trabalho

Este trabalho foi organizado da seguinte forma:

- O Capítulo 2 apresenta os fundamentos necessários para o desenvolvimento da proposta, abordando conceitos de Lógica Modal, e métodos para projeto de interfaces.
- O Capítulo 3 explica detalhadamente o método proposto, aplicando-o como exemplo em um sistema educativo, aqui nomeado de Portal de Ensino. Tal Portal é uma proposta de ferramenta para ensino de conteúdo para crianças, que, no entanto, pode ser estendida para o ensino de qualquer conteúdo.

-
- O Capítulo 4 apresenta um comparativo entre o método proposto e alguns métodos existentes na literatura para projeto de sistemas hipermídia.
 - O Capítulo 5 apresenta as conclusões do trabalho, com sugestões de trabalhos futuros.

Capítulo 2

Fundamentos

2.1 Fundamentos de lógica modal

A apresentação dos fundamentos da Lógica Modal, considerada a seguir, segue os três passos fundamentais: definição da linguagem, da semântica e do sistema de dedução.

2.1.1 Lógica Modal

Definição 2.1. (Linguagem da Lógica Modal)

A linguagem da Lógica Modal é um conjunto Φ de letras proposicionais (símbolos proposicionais ou variáveis proposicionais) cujos elementos são normalmente denotados por $p, q, r \dots$, e dois operadores unários $\Diamond$ ('ouro') e $\Box$ ('caixa') [Blackburn et al. 2002]. As fórmulas ϕ da linguagem da Lógica Modal são definidas pela forma de Backus Naur (BNF) a seguir:

$$\phi ::= p \mid \perp \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \phi \rightarrow \phi \mid \phi \leftrightarrow \phi \mid \Diamond\phi \mid \Box\phi$$

Conforme essa definição, uma fórmula proposicional pode ser tanto uma letra proposicional ou a constante de falsidade $\perp$.

Além disso, uma fórmula pode ser uma fórmula negada, uma disjunção de fórmulas, uma conjunção de fórmulas, uma implicação de fórmulas, uma equivalência de fórmulas, uma fórmula prefixada por um ouro $\Diamond$, ou por uma caixa $\Box$.

O conectivo ouro significa “possibilidade”, enquanto que o conectivo caixa tem o sentido de “necessidade”. Por exemplo, a fórmula

$$\Diamond\phi \tag{2.1}$$

é verdadeira em um estado, se, e somente se, em ao menos um estado alcançável de um estado atual, ϕ é verdadeiro. E a fórmula

$$\Box\phi \tag{2.2}$$

é verdadeira em um estado, se, e somente se, em todos os estados alcançáveis do estado atual, ϕ é verdadeiro.

Exemplo 2.1. Fórmulas da Lógica Modal

Exemplos de fórmulas da lógica modal são:

- $(p \wedge \Diamond(p \rightarrow \Box\neg r))$
- $\Box((\Diamond q \wedge \neg r) \rightarrow \Box p)$

Observa-se que é possível resumir a regra da linguagem modal para

$$\phi ::= p | \bot | \neg\phi | \phi \vee \phi | \Diamond\phi$$

Isso é obtido, levando-se em consideração as seguintes propriedades:

1. Os operadores $\Diamond$ e $\Box$ se comportam da mesma forma que os quantificadores Universal e Existencial, ou seja, as equivalências a seguir são válidas:

$$\forall x\alpha \leftrightarrow \neg\exists x\neg\alpha$$

$$\Box\phi \leftrightarrow \neg\Diamond\neg\phi$$

2. Também, analogamente à Lógica Proposicional, têm-se as equivalências:

- a. $\psi \wedge \phi \leftrightarrow \neg\psi \vee \neg\phi$
- b. $(\psi \rightarrow \phi) \leftrightarrow (\neg\psi \vee \phi)$
- c. $(\psi \leftrightarrow \phi) \leftrightarrow ((\psi \rightarrow \phi) \wedge (\phi \rightarrow \psi))$
- d. $\top \leftrightarrow \neg\bot$

A partir dessa linguagem, são utilizados *frames* para diagramar os modelos, ou interpretações das fórmulas do sistema.

Definição 2.2. (*Frames*)

Um *frame* é uma tupla $\mathfrak{F} = (W, R)$ tal que [Blackburn et al. 2002]:

- i. W é um conjunto não vazio, chamado de universo (ou domínio) de $\mathfrak{F}$
- ii. R é uma relação binária em W dita relação de acessibilidade (ou relação de visibilidade).

Os elementos de W recebem diversas designações, dependendo do contexto em que se apresentam. Exemplos dessas designações são:

- *mundos*,
- *pontos*,
- *estados*,
- *nós*,
- *instantes* e
- *situações*.

Como o objetivo da proposta apresentada nesta dissertação é a análise do fluxo navegacional de telas, então o termo “*tela*” é usado como sinônimo dos termos: *mundos*, *pontos*, *estados*, *nós*, *instantes* e *situações*.

Sendo $w_1, w_2 \in W$, diz-se que w_2 é acessível a partir de w_1 , ou ainda, w_2 possui uma relação de acessibilidade com w_1 , se $(w_1, w_2) \in R$ [Gouveia et al. 2000].

Portanto, um *frame* é simplesmente uma estrutura relacional com relações binárias simples.

Exemplo 2.2. *Frame*

A Figura 2.1 apresenta um exemplo de *frame*, $\mathfrak{F} = (W, R)$, tal que:

$$W = \{w_1, w_2, w_3, w_4\}$$

$$R = \{(w_1, w_2), (w_1, w_4), (w_2, w_4), (w_4, w_3), (w_3, w_1)\}$$

A semântica de um *frame* é definida como um modelo que determina os significados dos seus componentes.

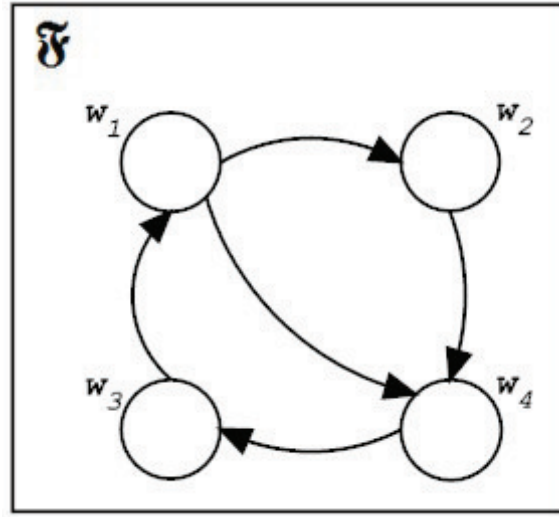


Figura 2.1: Frame

Definição 2.3. (Modelo)

Um modelo $\mathcal{M}$ da lógica modal é especificado por [Huth e Ryan 2008]:

1. Um conjunto W , cujos elementos são chamados mundos;
2. Uma Relação R em W tal que $R \subseteq W \times W$, chamada de relação de acessibilidade;
3. Uma função $V : W \rightarrow P$, tal que P é um conjunto de símbolos proposicionais, chamada de função de valoração.

Assim, um modelo $\mathcal{M}$ pode ser denotado:

$$\mathcal{M} = (W, R, V)$$

Observa-se que os conjuntos W e R definem uma estrutura relacional, definida como *Frame*. Desta forma, um modelo pode ser definido como:

$$\mathcal{M} = (\mathfrak{F}, V)$$

Assim, diz-se que $\mathcal{M}$ é um modelo baseado no frame $\mathfrak{F}$, ou que $\mathfrak{F}$ é o *frame* subjacente a $\mathcal{M}$. Além disso, a função V , chamada função de valoração, é o conjunto Φ de proposições que são verdadeiras nas telas de $\mathfrak{F}$ [Blackburn et al. 2002].

Definição 2.4. (Função de Valoração)

Seja $\mathcal{M} = (W, R, V)$ um modelo da lógica modal. Suponha que $w \in W$ e ϕ é uma fórmula da lógica modal. Define-se que uma fórmula ϕ é verdadeira em um mundo w

quando a fórmula ϕ é satisfeita no mundo w , ou seja:

$$w \Vdash \phi$$

Além disso, dada a função de valoração V , então $V(w)$ é o conjunto de todos os símbolos proposicionais que são verdadeiros em w [Huth e Ryan 2008].

Definição 2.5. (Satisfação de Fórmulas em Mundos de um Modelo)

Seja w um mundo no modelo

$$\mathcal{M} = (W, R, V)$$

Uma fórmula ϕ é satisfeita (verdadeira) em $\mathcal{M}$, no mundo w se, e somente se: $\mathcal{M}, w \Vdash \phi$,

$w \Vdash p$ se, e somente se, $p \in V(w)$.

$\mathcal{M}, w \not\Vdash \perp$.

$\mathcal{M}, w \Vdash \neg\phi$ se, e somente se, $\mathcal{M}, w \not\Vdash \phi$.

$\mathcal{M}, w \Vdash \phi \vee \psi$ se, e somente se, $\mathcal{M}, w \Vdash \phi$ ou $\mathcal{M}, w \Vdash \psi$.

$\mathcal{M}, w \Vdash \phi \wedge \psi$ se, e somente se, $\mathcal{M}, w \Vdash \phi$ e $\mathcal{M}, w \Vdash \psi$.

$\mathcal{M}, w \Vdash \phi \rightarrow \psi$ se, e somente se, $\mathcal{M}, w \Vdash \phi$ sempre que $\mathcal{M}, w \Vdash \psi$.

$\mathcal{M}, w \Vdash \phi \leftrightarrow \psi$ se, e somente se, $\mathcal{M}, w \Vdash \phi$ se, e somente se, $\mathcal{M}, w \Vdash \psi$.

$\mathcal{M}, w \Vdash \Box\phi$ se, e somente se, para todo $v \in W$ tal que $R(w, v)$, temos $\mathcal{M}, v \Vdash \phi$.

$\mathcal{M}, w \Vdash \Diamond\phi$ se, e somente se, existe um $v \in W$ tal que $R(w, v)$, temos $\mathcal{M}, v \Vdash \phi$.

Além disso, um conjunto Σ de fórmulas é verdadeiro em um estado w de um modelo $\mathcal{M}$, denotado por:

$$\mathcal{M}, w \Vdash \Sigma$$

se, e somente se, todos os membros de Σ são verdadeiros em w .

Definição 2.6. (Satisfação de Fórmulas em um Modelo)

Um modelo $\mathcal{M} = (W, R, V)$ satisfaz a uma fórmula se em todos os mundos do modelo a fórmula é satisfeita [Huth e Ryan 2008]. Assim:

$$\mathcal{M} \Vdash \phi$$

se, e somente se, para cada $w \in W$, $w \Vdash \phi$.

Exemplo 2.3. Modelo

A Figura 2.2 apresenta um exemplo de modelo, $\mathcal{M} = (W, R, V)$, tal que:

$$W = \{w_1, w_2, w_3, w_4\}$$

$$R = \{(w_1, w_2), (w_1, w_4), (w_2, w_4), (w_4, w_3), (w_3, w_1)\}$$

e a função de valoração V aplicada em cada mundo da seguinte forma:

$$\begin{aligned} V(w_1) &= \{p, q\}; \\ V(w_2) &= \{p, r\}; \\ V(w_3) &= \{p, q, s\}; \\ V(w_4) &= \{p\}. \end{aligned}$$

Pode-se concluir que as seguintes fórmulas são satisfeitas em $\mathcal{M}$:

- $\mathcal{M}, w_4 \Vdash p$,
pois $p \in V(w_4)$.
- $\mathcal{M}, w_1 \Vdash \neg r$,
pois $\mathcal{M}, w_1 \nVdash r$, ou seja, r não está presente em w_1 .
- $\mathcal{M}, w_3 \Vdash p \wedge q \wedge s$,
pois $\mathcal{M}, w_3 \Vdash p$, $\mathcal{M}, w_3 \Vdash q$ e $\mathcal{M}, w_3 \Vdash s$.
- $\mathcal{M}, w_3 \Vdash p \wedge q \wedge s \wedge \neg r$,
pois $\mathcal{M}, w_3 \Vdash p$, $\mathcal{M}, w_3 \Vdash q$, $\mathcal{M}, w_3 \Vdash s$ e $\mathcal{M}, w_3 \nVdash r$.
- $\mathcal{M}, w_1 \Vdash \Box p$,
pois todos os mundos acessíveis de w_1 satisfazem p .
- $\mathcal{M}, w_1 \nVdash \Box r$,
porque existe um mundo acessível de w_1 que não satisfaz r (a saber, w_4).
- $\mathcal{M}, w_1 \Vdash \Diamond r$,
pois existe um mundo acessível de w_1 que satisfaz r , o mundo w_2 .
- $\mathcal{M} \Vdash p$,
significa que em todos os mundos do modelo $\mathcal{M}$, p é válido.

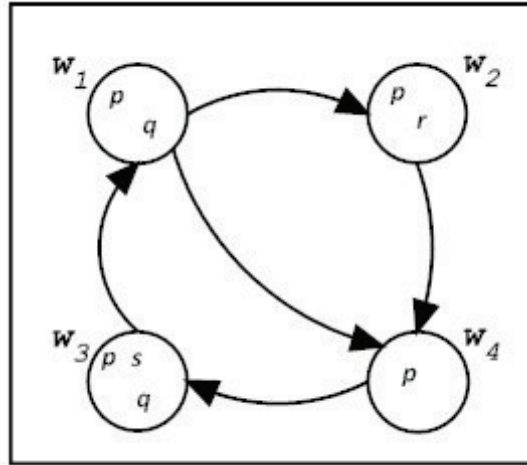


Figura 2.2: Modelo

Definição 2.7. (Caminho)

Um Caminho em um modelo $\mathcal{M} = (W, R, V)$ é uma sequência infinita de mundos $w_1, w_2, w_3, \dots \in W$, tal que, para cada $i \geq 1$, $(w_i, w_{i+1}) \in R$. Um caminho é escrito da seguinte forma:

$$\pi = w_1 \rightarrow w_2 \rightarrow w_3 \rightarrow \dots$$

Notação: Um caminho é representado pela letra grega π . O trecho final de um caminho, iniciado no mundo w_i , é designado de π^i .

Exemplo 2.4. Caminho

Considere o modelo do Exemplo 2.3, representado pela Figura 2.2. O caminho

$$\pi = w_1 \rightarrow w_4 \rightarrow w_3 \rightarrow \dots$$

representa um incremento linear do tempo.

O caminho inicia no mundo w_1 , depois passa ao mundo w_4 , depois ao mundo w_3 , e assim por diante.

O trecho do caminho π , iniciado no mundo w_4 , ou seja, no segundo estado do caminho, é

$$\pi^2 = w_4 \rightarrow w_3 \rightarrow \dots$$

E o trecho iniciado no mundo w_3 é

$$\pi^3 = w_3 \rightarrow \dots$$

No modelo navegacional proposto neste trabalho, as propriedades de um sistema de telas são verificadas considerando os fundamentos da dedução semântica da Lógica Modal. A seguir, são apresentados alguns exemplos de como verificar a satisfação de propriedades, escritas como fórmulas da lógica modal em um modelo. Mais adiante, neste trabalho, tais verificações são utilizadas para analisar propriedades navegacionais em um sistema de telas.

Exemplo 2.5. Uso dos Operadores $\Diamond$ e $\Box$

A Figura 2.3 apresenta um modelo $\mathcal{M} = (W, R, V)$, com:

$$W = \{w_1, w_2, w_3, w_4\}$$

$$R = \{(w_1, w_2), (w_1, w_3), (w_1, w_4)\}$$

Nesse modelo, V é tal que:

$$V(w_1) = \{p, q\};$$

$$V(w_2) = \{p, q\};$$

$$V(w_3) = \{q\};$$

$$V(w_4) = \{q\};$$

Em um caminho iniciado em w_1 , conclui-se que:

- $\mathcal{M}, w_1 \Vdash \Diamond p$

é válida, pois existe um caminho,

$$\pi = w_1 \rightarrow w_2$$

de forma que no segundo mundo da sequência, w_2 , p é satisfeito.

- $\mathcal{M}, w_1 \Vdash \Box q$

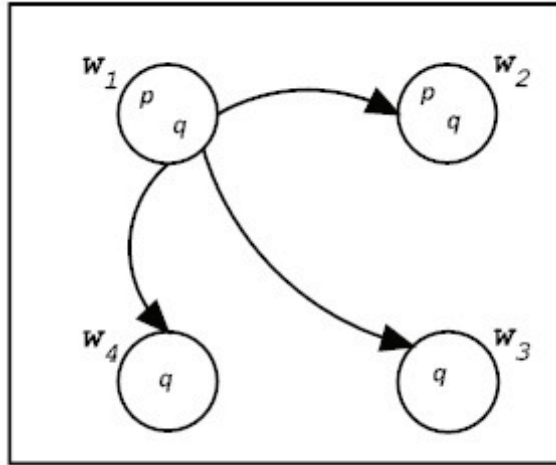
é válida, pois em todos os caminhos iniciados em w_1 , os quais são:

$$\pi_1 = w_1 \rightarrow w_2$$

$$\pi_2 = w_1 \rightarrow w_3$$

$$\pi_3 = w_1 \rightarrow w_4$$

no segundo mundo desses caminhos, q é satisfeito.

Figura 2.3: Operadores $\Diamond$ e $\Box$

Exemplo 2.6. Transitividade do Operador $\Diamond$

A Figura 2.4 apresenta um modelo $\mathcal{M} = (W, R, V)$, com:

$$W = \{w_1, w_2, w_3, w_4\}$$

$$R = \{(w_1, w_2), (w_1, w_3), (w_1, w_4), (w_2, w_3)\}$$

Nesse modelo, V é tal que:

$$V(w_1) = \{p\};$$

$$V(w_2) = \{p\};$$

$$V(w_3) = \{p\};$$

$$V(w_4) = \{\};$$

Em um caminho iniciado em w_1 , conclui-se que:

$$\mathcal{M}, w_1 \Vdash \Diamond\Diamond p$$

é válida, pois existe um caminho,

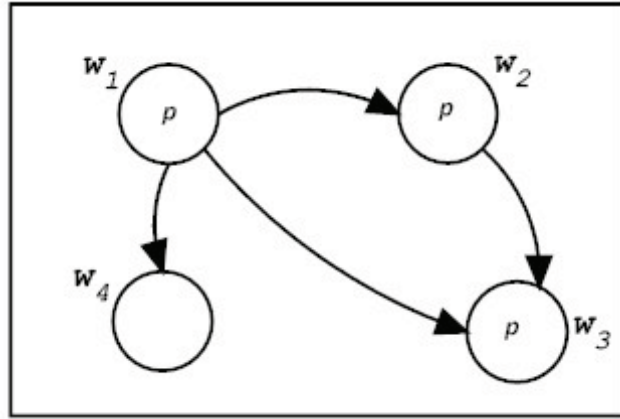
$$\pi = w_1 \rightarrow w_2 \rightarrow w_3$$

de forma que no terceiro mundo da sequência, w_3 , p é satisfeito.

Exemplo 2.7. Transitividade dos Operadores $\Diamond\Box$

A Figura 2.5 apresenta um modelo $\mathcal{M} = (W, R, V)$, com:

$$W = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10}\}$$

Figura 2.4: Transitividade do Operador $\Diamond$

$$R = \{(w_1, w_2), (w_1, w_3), (w_1, w_4), (w_2, w_5), (w_3, w_6), (w_3, w_7), (w_3, w_8), (w_4, w_9), (w_4, w_{10})\}$$

Nesse modelo, V é tal que:

$$\begin{aligned} V(w_1) &= \{p\}; \\ V(w_2) &= \{p\}; \\ V(w_3) &= \{p\}; \\ V(w_4) &= \{p\}; \\ V(w_5) &= \{p\}; \\ V(w_6) &= \{p\}; \\ V(w_7) &= \{\}; \\ V(w_8) &= \{\}; \\ V(w_9) &= \{\}; \\ V(w_{10}) &= \{p\} \end{aligned}$$

Em um caminho iniciado em w_1 , pode-se verificar que:

- $M, w_1 \models \Box \Diamond p$

é válida, pois a partir de todos os estados alcançáveis de w_1 , é possível traçar ao menos um caminho cujo próximo estado contém p . Têm-se, então, os caminhos:

$$\pi_1 = w_1 \rightarrow w_2 \rightarrow w_5$$

$$\pi_2 = w_1 \rightarrow w_3 \rightarrow w_6$$

$$\pi_3 = w_1 \rightarrow w_4 \rightarrow w_{10}$$

Os caminhos

$$\pi_4 = w_1 \rightarrow w_3 \rightarrow w_7$$

$$\pi_5 = w_1 \rightarrow w_3 \rightarrow w_8$$

$$\pi_6 = w_1 \rightarrow w_4 \rightarrow w_9$$

não satisfazem a fórmula $\Box\Diamond p$.

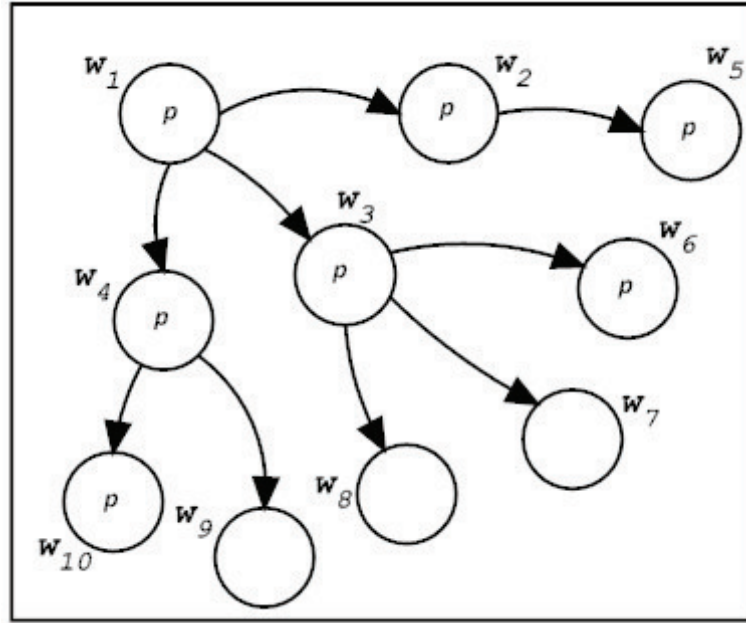


Figura 2.5: Transitividade do Operador $\Box\Diamond$

Exemplo 2.8. Transitividade dos Operadores: $\Diamond\Box$

A Figura 2.6 apresenta um modelo $\mathcal{M} = (W, R, V)$, com:

$$W = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10}\}$$

$$R = \{(w_1, w_2), (w_1, w_3), (w_1, w_4), (w_2, w_5), (w_3, w_6), (w_3, w_7), (w_3, w_8), (w_4, w_9), (w_4, w_{10})\}$$

Nesse modelo, V é tal que:

$$V(w_1) = \{p\};$$

$$V(w_2) = \{\};$$

$$V(w_3) = \{p\};$$

$$V(w_4) = \{\};$$

$$\begin{aligned}
V(w_5) &= \{\}; \\
V(w_6) &= \{p\}; \\
V(w_7) &= \{p\}; \\
V(w_8) &= \{p\}; \\
V(w_9) &= \{\}; \\
V(w_{10}) &= \{\}
\end{aligned}$$

Em um caminho iniciado em w_1 , pode-se verificar que:

- $\mathcal{M}, w_1 \models \Diamond \Box p$

é válida, pois a partir de ao menos um estado alcançável de w_1 , em todos os caminhos p ocorre no estado seguinte. Ou seja, no modelo da Figura 2.6 os caminhos que satisfazem a fórmula $\Diamond \Box p$ são:

$$\pi_1 = w_1 \rightarrow w_3 \rightarrow w_6$$

$$\pi_2 = w_1 \rightarrow w_3 \rightarrow w_7$$

$$\pi_3 = w_1 \rightarrow w_3 \rightarrow w_8$$

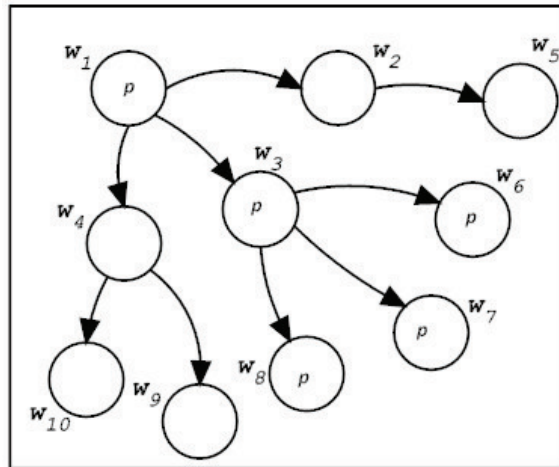


Figura 2.6: Transitividade do Operador $\Diamond \Box$

A proposta apresentada neste trabalho considera um fragmento da Lógica Modal denominado Lógica Temporal. Lógicas temporais tem um aspecto dinâmico, pois a verdade de uma fórmula não é fixa em um modelo, como ocorre na lógica proposicional e de predicados, mas depende de um ponto no tempo dentro do modelo [Huth e Ryan 2008].

2.1.2 Lógica Temporal Linear

Entre os vários tipos de Lógica Temporal, a proposta apresentada neste trabalho considera a Lógica Temporal de Tempo-Linear (LTL) e a Lógica de Árvore Computacional (LAC).

Observa-se que tais escolhas podem ser consideradas como opções de projeto. Outros tipos de Lógicas Temporais poderiam ser escolhidos e, nesse caso, os modelos obtidos teriam propriedades diferentes. Neste trabalho considera-se a LTL e a LAC, pelo sua expressividade adequada para representação de fluxo.

Ao final da apresentação da proposta, esse tema de análise será considerado novamente.

Definição 2.8. (Lógica Temporal Linear)

A Lógica Temporal Linear, LTL, modela o tempo como uma sequência de estados, estendendo infinitamente para o futuro. Dessa forma, o tempo não é determinístico, podendo ocorrer diversos caminhos na sequência temporal, o que possibilita a representação de diferentes futuros possíveis. As fórmulas da LTL possuem a seguinte sintaxe: [Huth e Ryan 2008]

$$\phi ::= \top | \perp | p | (\neg \phi) | (\phi \wedge \phi) | (\phi \vee \phi) | (\phi \rightarrow \phi) | (\phi \leftrightarrow \phi) | (X\phi) | (F\phi) | (G\phi) | (\phi U \phi) | (\phi W \phi) | (\phi R \phi)$$

Os conectivos X , F , G , U , W e R são chamados conectivos temporais e tem os seguintes significados:

- X significa “próximo estado” (*neXt state*),
- F significa “em algum estado Futuro” (*some Future state*),
- G significa “todos os estados futuros” (*Globally*),
- U significa “até que” (*Until*),
- R significa “libertação” (*Release*) e
- W significa “um ‘até que’ fraco” (*Weak-until*).

Sistemas verificados por LTL podem ser modelados como sistemas de transição, ou seja, sistemas compostos por estados e transições. Nesse caso, tem-se exatamente o que ocorre na análise navegacional de um sistema de telas.

Definição 2.9. (A Semântica da Lógica Temporal Linear)

Na LTL, um modelo de transição $\mathcal{M}$, é uma tupla [Huth e Ryan 2008]

$$\mathcal{M} = (S, \rightarrow, L)$$

tal que S é um conjunto de estados ou mundos dotados de uma relação de transição $\rightarrow$. O conjunto S possui o mesmo conceito que o conjunto W da Lógica Modal. A relação de transição $\rightarrow$ é uma relação binária em S , semelhante ao conjunto R da Lógica Modal. Além disso, para cada $s \in S$ existe um $s' \in S$ tal que,

$$s \rightarrow s'$$

A função de rotulação L , funciona da mesma forma que a função de valoração V da Lógica Modal:

$$L : S \rightarrow P$$

Definição 2.10. (Satisfação de fórmulas da LTL)

Seja

$$\mathcal{M} = (S, \rightarrow, L)$$

um modelo de transição e

$$\pi = s_1 \rightarrow \dots$$

um caminho em $\mathcal{M}$.

O caminho π satisfaz uma fórmula LTL, se satisfaz alguma das relações a seguir, [Huth e Ryan 2008]

i. $\pi \models \top$

ii. $\pi \not\models \perp$

iii. $\pi \models p$ se, e somente se, $p \in L(s_1)$

iv. $\pi \models \neg\phi$ se, e somente se, $\pi \not\models \phi$

v. $\pi \models \phi_1 \wedge \phi_2$ se, e somente se, $\pi \models \phi_1$ e $\pi \models \phi_2$

- vi. $\pi \models \phi_1 \vee \phi_2$ se, e somente se, $\pi \models \phi_1$ ou $\pi \models \phi_2$
- vii. $\pi \models \phi_1 \rightarrow \phi_2$ se, e somente se, $\pi \models \phi_1$ sempre que $\pi \models \phi_2$
- viii. $\pi \models \phi_1 \leftrightarrow \phi_2$ se, e somente se, $\pi \models \phi_1 \rightarrow \phi_2$ e $\pi \models \phi_2 \rightarrow \phi_1$
- ix. $\pi \models X\phi$ se, e somente se, $\pi^2 \models \phi$
- x. $\pi \models G\phi$ se, e somente se, para todo $i \geq 1$, $\pi^i \models \phi$
- xi. $\pi \models F\phi$ se, e somente se, existe $i \geq 1$, tal que $\pi^i \models \phi$
- xii. $\pi \models \phi_1 U \phi_2$ se, e somente se, existe $i \geq 1$, tal que $\pi^i \models \phi_2$ e para todo $j = 1, \dots, i-1$, temos que $\pi^j \models \phi_1$
- xiii. $\pi \models \phi_1 W \phi_2$ se, e somente se, existe $i \geq 1$, tal que $\pi^i \models \phi_2$ e para todo $j = 1, \dots, i-1$, temos que $\pi^j \models \phi_1$; ou para todo $k \geq 1$ temos $\pi^k \models \phi_1$
- xiv. $\pi \models \phi_1 R \phi_2$ se, e somente se, existe $i \geq 1$, tal que $\pi^i \models \phi_1$ e para todo $j = 1, \dots, i-1$, temos que $\pi^j \models \phi_2$; ou para todo $k \geq 1$ temos $\pi^k \models \phi_2$

Exemplo 2.9. O Operador X

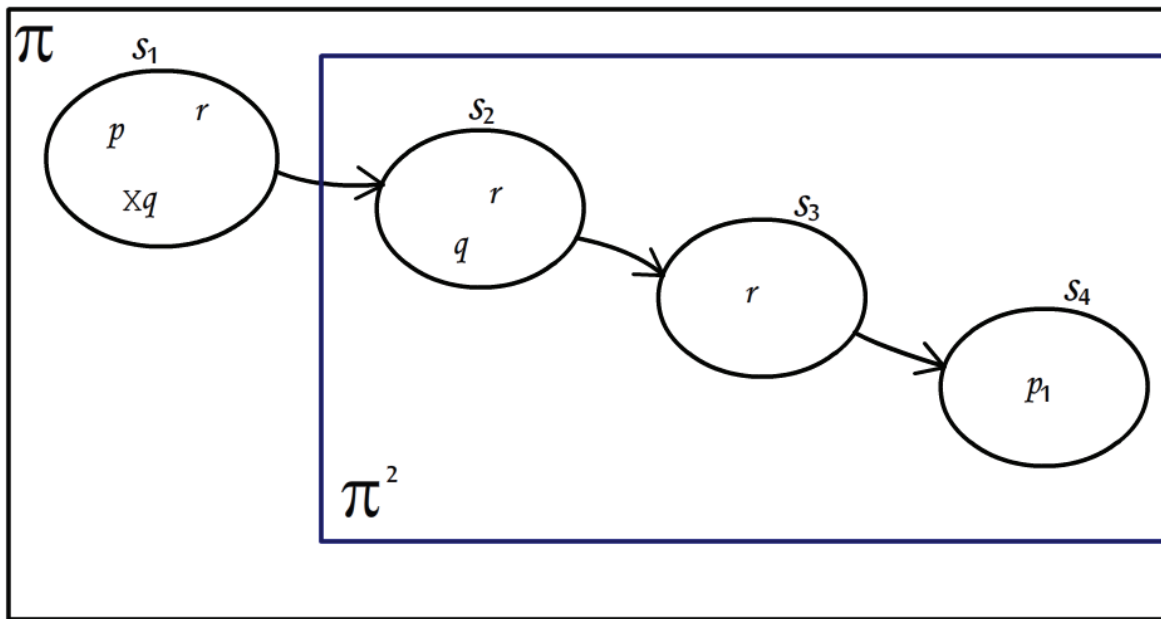
A Figura 2.7 apresenta um caminho π de um modelo, e, também, exemplifica a semântica do operador X . Nesse caminho, a fórmula

$$s_1 \models Xq$$

é verdadeira, pois o segundo estado do caminho (estado s_2) contém q . Ou seja, a partir do caminho π , tem-se o caminho π^2 , cujo primeiro mundo satisfaz q .

Exemplo 2.10. O Operador G

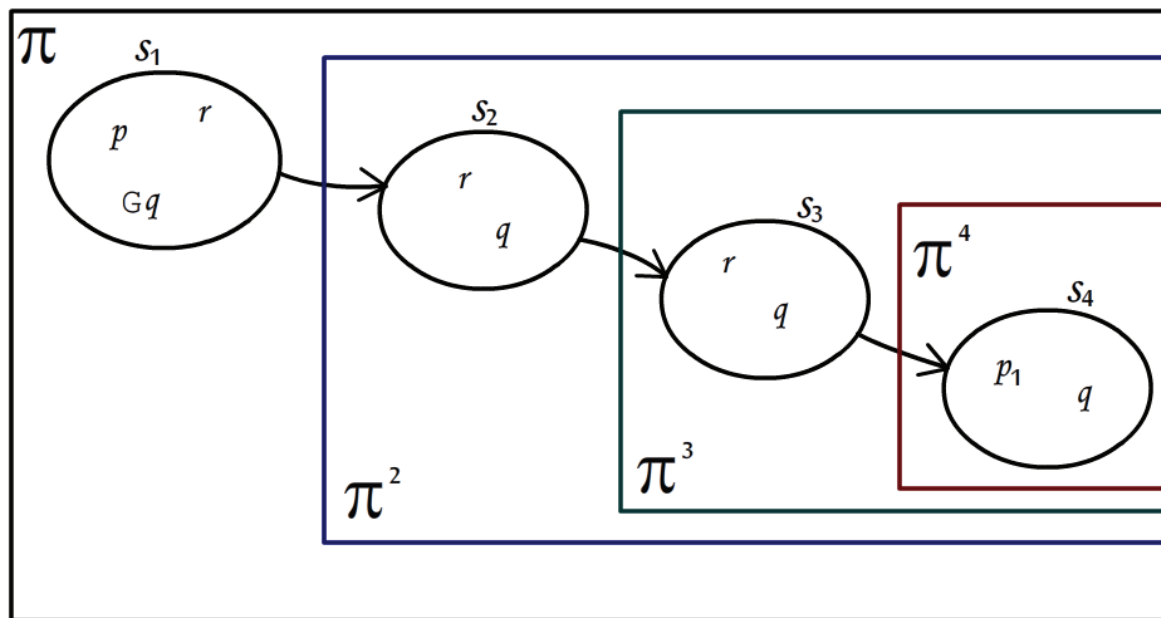
A Figura 2.8 apresenta um exemplo da semântica do operador G . A Figura mostra

Figura 2.7: Exemplo do operador X

que a fórmula

$$s_1 \models Gq$$

é verdadeira no caminho π , pois, a partir do segundo estado q ocorre em todos os mundos. Ou seja, q está contido em todos os mundos do caminho π^2 .

Figura 2.8: Exemplo do operador G

Exemplo 2.11. O Operador F

O operador F é satisfeito se, estando em um mundo s_1 , em algum momento no futuro q vai ocorrer, ou seja, q é satisfeito em algum mundo do caminho π iniciado no estado s_1 . A Figura 2.9 exemplifica a semântica desse operador. Nesse caso,

$$s_1 \models Fq$$

é satisfeita, porque em um momento do caminho iniciado por s_1 ocorre q , e esse momento é o mundo s_3 do caminho π .

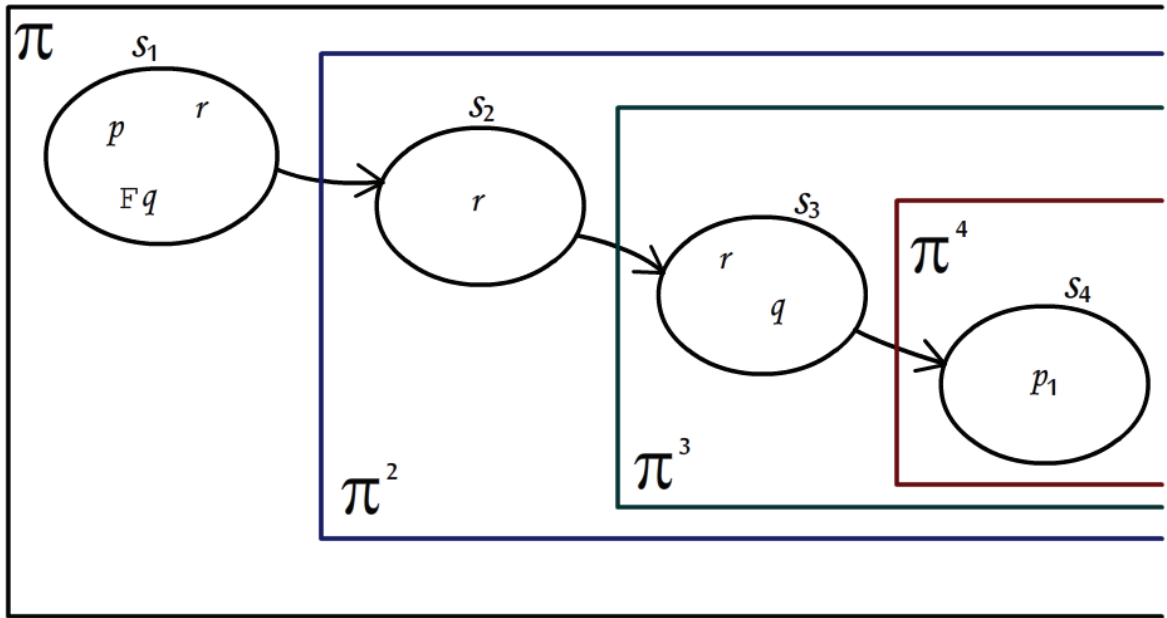


Figura 2.9: Exemplo do operador F

Exemplo 2.12. Os Operadores U e W

A Figura 2.10 demonstra a semântica do operador U . Quando U é utilizado em uma dedução do tipo

$$s_1 \models pUq$$

significa que p vai ocorrer em todos os mundos do caminho π até o momento que q ocorrer. Ou seja, no caminho π^i , onde q aparece pela primeira vez, p passa a estar desativado e não ocorre em nenhum dos outros mundos desse caminho.

O operador W tem semântica similar. A dedução

$$s_1 \models pWq$$

também é verdadeira para o caminho π , a única diferença é que q poderia não ocorrer, de forma que p ficaria ativado em todos os estados do caminho π .

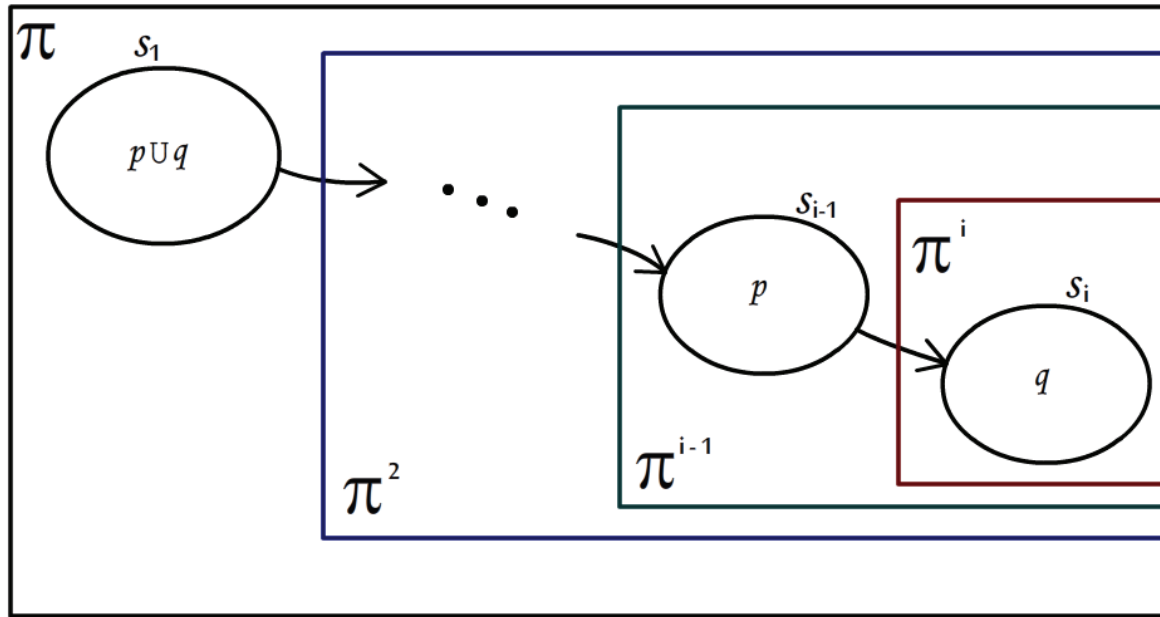


Figura 2.10: Exemplo do operador U

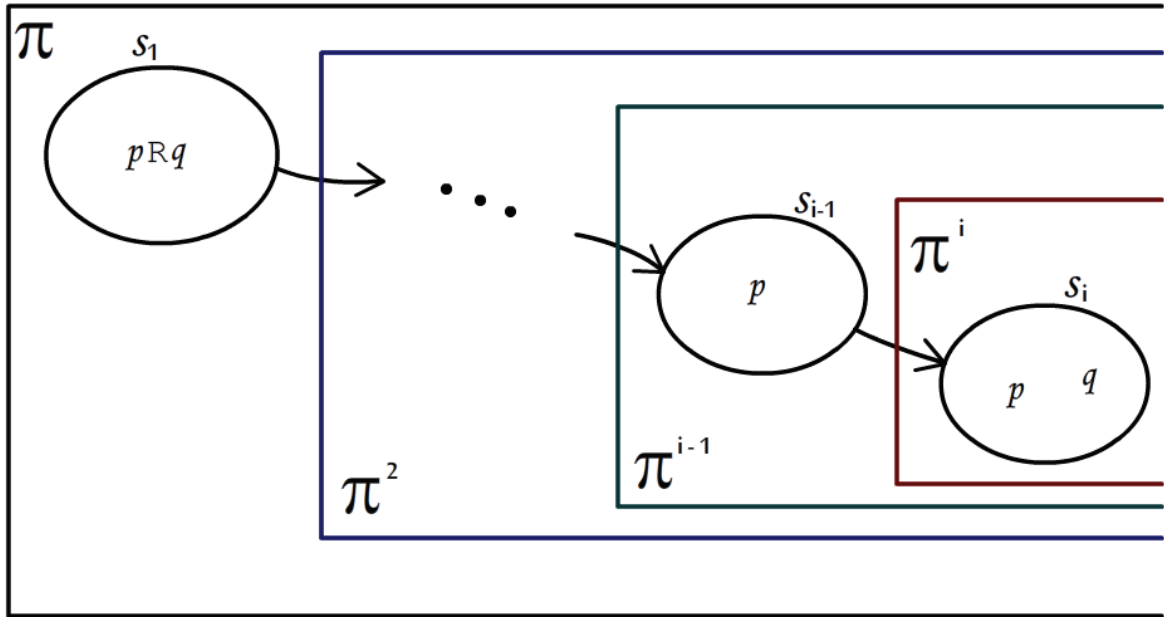
Exemplo 2.13. O Operador R

A semântica do operador R é mostrada na Figura 2.11. A satisfação da dedução

$$s_1 \models pRq$$

aplicada no primeiro mundo do caminho π , é satisfeita na Figura 2.11. Isso, porque p ocorre em todos os mundos do caminho, até o momento que q ocorre, inclusive. Ou seja, no primeiro mundo do caminho π^i , q é ativado, fazendo com que a partir do primeiro mundo do caminho $\pi^{(i+1)}$, p fique desativado.

Um dos fundamentos da semântica da Lógica Temporal Linear, LTL, é o estudo de caminhos temporais para os quais uma fórmula é, ou não, satisfeita. Assim, uma fórmula pode ser satisfeita, ou não, em todos os caminhos a partir de determinado estado, onde ela está sendo testada.

Figura 2.11: Exemplo do operador R

No contexto da LTL não é possível analisar a satisfação de uma fórmula em apenas um caminho em separado. Nesse sentido, a LTL não é uma Lógica Temporal Ramificada, que considera esse tipo de problema, ou seja, a verificação de uma fórmula em algum caminho temporal, mas não em todos. Ou, também, a verificação da fórmula em todos os caminhos.

2.1.3 Lógica de Árvore Computacional

Na proposta ora apresentada é também analisada a utilização da Lógica Ramificada, denominada por Lógica de Árvore Computacional (LAC), que além dos conectivos da sintaxe da LTL, possui outros dois conectivos: A e E. Tais conectivos significam, respectivamente, “todos os caminhos” e “existe um caminho”. Essa abordagem é adequada para a representação de fluxo de navegação, pois são vários os caminhos possíveis que o usuário pode percorrer ao interagir com uma interface, e cada um desses caminhos pode alcançar uma área ou função diferente do sistema, em momentos diferentes.

Definição 2.11. (A Sintaxe da LAC)

As fórmulas da LAC possuem a seguinte sintaxe: [Huth e Ryan 2008]

$$\phi ::= \perp \mid \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid AX\phi \mid EX\phi \mid AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid \phi AU\phi \mid \phi EU\phi$$

Na Lógica de Árvore Computacional, além dos conectivos usuais da lógica, há outros formados por pares de símbolos. O primeiro símbolo do par pode ser A ou E . Nesse caso, A significa “em todos os caminhos” (*along All paths*) (inevitabilidade) e E significa “existe um caminho” (*there Exists*) (possibilidade). O segundo símbolo é um dos conectivos X , F , G ou U da Lógica Temporal Linear, LTL. Os conectivos W e R não são incluídos na LAC, mas podem ser facilmente derivados a partir dos demais conectivos.

Na proposta apresentada, os métodos de verificação são aqueles definidos pelos sistemas de dedução da Lógica Modal. Assim, no contexto da proposta, os métodos de verificação podem ser utilizados para verificar se a descrição do sistema satisfaz a especificação.

2.2 Métodos de Modelagem de Interface

Definição 2.12. (Interface)

A interface de usuário deve ser entendida como sendo a parte de um sistema computacional com o qual uma pessoa entra em contato física, perspectiva e conceitualmente [?].

Segundo o disposto na Definição 2.12, a interface do usuário de um sistema computacional é a parte que exhibe ao usuário o que o programa faz, como ele pode utilizá-lo, quais informações devem ser fornecidas, normalmente utilizando janelas, menus, ícones, botões. [Orth 2005].

Desenvolver uma interface usável é imprescindível para o sucesso de um software [Orth 2005]. Uma interface usável é criada quando as necessidades de um usuário são atendidas sem atrapalhá-lo na realização de suas tarefas. Usuários não estão interessados em utilizar a interface e sim resolver, mediante o uso do computador, um problema qualquer. Quanto menos a interface for notada pelo usuário, melhor [Orth 2005].

Projetar uma interface é uma atividade complexa e crucial no desenvolvimento de um software. Estima-se que de 60% a 90% de todas as falhas de softwares sejam atribuídas a problemas na interação entre os sistemas e seus usuários [Hollnagel 1993].

O objetivo do projeto de interface é criar uma interface usável, no entanto, uma interface usável não faz um bom programa. Nenhuma aplicação ruim ou inadequada poderá ser melhorada por meio de uma interface usável [Mandel 1997].

Dessa forma, a maior dificuldade, atualmente, é conciliar o projeto da interface do usuário, com a modelagem e desenvolvimento do sistema em si. Decisões referentes ao projeto da interface do usuário têm profundas implicações no projeto e desenvolvimento do sistema como um todo, não apenas na implementação da interface do usuário [Campos 2004].

Diversas técnicas foram desenvolvidas para garantir o projeto de uma interface usável, e conseqüentemente melhorar a relação entre o projeto do sistema com o projeto da interface.

Definição 2.13. (Especificação de Interface)

Uma especificação é uma descrição detalhada de como algo pode ser feita ou construída [Silva 2007]. A especificação de um software destina-se a estabelecer quais funções são requeridas pelo sistema e as restrições sobre as operações e o desenvolvimento do sistema [Sommerville 2007].

Com base nessa definição, pode-se concluir que a Especificação de uma Interface é a definição da forma de interação que o usuário terá ao operar a interface, ou seja, o conjunto de operações que ele poderá realizar, como será a interação do usuário com o sistema, e como o sistema responderá a cada ação do usuário.

2.2.1 Métodos de Especificação de Interface

A seguir são apresentados alguns exemplos de métodos e linguagens que são utilizados para especificação de interfaces.

Exemplo 2.14. Backus-Naur Form

A *Backus-Naur Form*, BNF, é uma notação para modelos gramaticais, que são conjuntos de regras que definem a sintaxe de uma determinada linguagem. A BNF consiste de operadores, nomes terminais e não terminais [Silva 2007].

Os símbolos terminais representam as ações possíveis do usuário em um sistema, como pressionar um botão, inserir um texto, e os símbolos terminais são definidos através da

combinação de terminais e não terminais. A interação usuário-sistema é especificada através das regras [Silva 2007].

Exemplo 2.15. *Task-Action Grammars*

A *Task-Action Grammars*, TAG, é uma variação da BNF que utiliza regras gramaticais parametrizadas para enfatizar a consistência e o conhecimento do usuário sobre as nomenclaturas do seu ambiente [Payne e Green 1989].

A TAG foca na interação do sistema, e baseia-se nas ações do usuário.

Exemplo 2.16. *Command Language Grammar*

A *Command Language Grammar*, CLG, provê um *framework* para organizar decisões de design [?]. O *framework* é um conjunto ordenado de descrições de níveis diferentes de abstração.

A GLC engloba as tarefas do usuário, os objetos do sistema e como manipulá-los, a sintaxe da interação, com o diálogo envolvido na utilização do sistema.

Exemplo 2.17. *State Transition Diagrams*

Os *State Transition Diagrams*, STD, são usados como uma representação gráfica do fluxo do controle da interação de forma sequencial, tendo como principal ponto os estados do sistema durante a interação [Parnas 1969].

O STD é formado de estados e transições que são disparadas através de ações do usuário, causando a mudança de estado.

Exemplo 2.18. *Statechart*

Os *Statecharts* estendem o formalismo do STD e são caracterizados por uma estrutura hierárquica, com a adição da possibilidade de existirem estados alternativos e concorrentes [Harel 1987].

Exemplo 2.19. *Goals, Operators, Methods and Selection Rules*

O método *Goals, Operators, Methods and Selection Rules*, GOMS, é definido por [Kieras 1999]:

- Metas (*Goals*) - representa algo que o usuário deseja atingir.

- Operadores (*Operators*) - ações do usuário para alcançar uma meta.
- Métodos (*Methods*) - sequência de passos para realizar uma tarefa, um passo é um conjunto de operadores para realizar uma meta.
- Regras de seleção (*Selection Rules*) - serve para determinar qual método, dentre os métodos existentes para uma mesma meta, é o mais apropriado.

Exemplo 2.20. *User Action Notation*

A *User Action Notation*, UAN, descreve o comportamento do usuário e da interface, e a forma que eles realizam uma tarefa em conjunto. As principais características representadas são as ações do usuário e o *feedback* da interface [Hix e Hartson. 1993].

Exemplo 2.21. *Abstract User Interface Markup Language*

A *Abstract User Interface Markup Language*, AUIML, é um vocabulário XML que foi projetado para permitir a especificação de uma interação de um usuário [Azevedo et al. 2000].

A AUIML utiliza tipos de dados simples para descrever ações, elementos de metadado e estruturas.

Exemplo 2.22. *User Interface Markup Language*

A *User Interface Markup Language*, UIML, também baseada em XML, expressa interfaces de usuário para múltiplas plataformas de *software* em diferentes dispositivos e para múltiplas aplicações [Phanouriou 2000].

Um documento UIML descreve uma representação da interface apropriada para um determinado dispositivo.

Exemplo 2.23. *eXtensible Interface Markup Language*

A *eXtensible Interface Markup Language*, XIML, baseada em XML, define um *framework* para a definição e relacionamento entre elementos de interação [Puerta e Eisenshtein 2001].

A linguagem XIML possui três principais unidades [Silva 2007]:

- Componentes - elementos da interface como: tarefas do usuário, objetos do domínio, tipos do usuário, elementos de apresentação e elementos de diálogo.

- Relação - declaração que associa dois ou mais componentes
- Atributos - propriedades de componentes que podem receber um valor.

Exemplo 2.24. *USer Interface eXtensible Markup Language*

A *USer Interface eXtensible Markup Language*, USIXML, descreve interfaces de usuário em vários níveis de detalhamento e abstração, dependendo do contexto do uso [Limbourg et al. 2004].

De acordo com [Silva 2007], os níveis são:

- Interface de Usuário Final, IUF - uma interface de usuário implementada e que possa ser executada.
- Interface de Usuário Concreta, IUC - abstração da IUF, independente de plataforma e com aparência definida.
- Interface de Usuário Abstrata, IUA - abstrai a IUC em uma definição independente da forma de interação.
- Tarefas - tarefa interativa a ser executada pelo usuário.
- Conceitos - objetos manipulados pela tarefa.

Exemplo 2.25. *Unified Modeling Language*

A *Unified Modeling Language*, UML, é a linguagem padrão para especificar e documentar software. Alguns de seus diagramas permitem especificar interfaces de usuário, tais quais, o Diagrama de Estados, o Diagrama de Atividades e o Diagrama de Sequência.

Exemplo 2.26. *Hypermedia Design Model*

O *Hypermedia Design Model*, HDM, foi um dos primeiros métodos que surgiram para descrever aplicações hipermídia [Baba 2005].

“HDM prescreve a definição de um esquema de aplicação, que descreve classes globais de elementos de informação em termos de suas características de apresentação comuns, suas estruturas de organização interna, e os tipos de interconexões mútuas.” [Garzotto e Paolini 1993]

Exemplo 2.27. *Object-Oriented Hypermedia Design Method*

O método *Object-Oriented Hypermedia Design Method*, OOHDM, é utilizado no desenvolvimento de aplicações hipermídia e o processo compreende quatro atividades que executam uma mistura de estilos iterativos e incrementais de desenvolvimento [Rossi 1996].

O método é composto de quatro atividades [Baba 2005]:

- Modelagem de domínio ou conceitual - usado para representar os objetos e os relacionamentos existentes no domínio.
- Projeto navegacional - as aplicações são definidas como visões navegacionais do esquema conceitual, e essas visões definem contextos e classes navegacionais.
 - Contexto Navegacional - representa a estrutura navegacional geral de uma aplicação.
 - Classe Navegacional - especifica os objetos que serão vistos pelo usuário
- Projeto da interface abstrata - define a estrutura navegacional que pode ser vista pelo usuário através da definição de um modelo de interface abstrata, ou seja, define os objetos perceptíveis pelo usuário.
- Implementação - mapeia os modelos de interface abstrata e navegacional de forma a gerar a aplicação pronta para ser executada.

Exemplo 2.28. *Hypermedia Modeling Technique*

A *Hypermedia Modeling Technique*, HMT, é um método orientado a objetos utilizado para projeto de aplicações multimídia.

Possui quatro fases para descrição de uma aplicação [Nemetz 1995]:

- Modelo de Objetos - descreve os objetos do domínio da aplicação e seus relacionamentos.
- Modelo de Hiperobjetos - refina o modelo de objetos, adicionando semântica aos relacionamentos.
- Modelo de Navegação - descreve os elos e as estruturas de acesso.
- Modelo de Interface - descreve como o usuário perceberá os objetos hipermídia.

Exemplo 2.29. *Modeling Language for Interaction as Conversation*

A *Modeling Language for Interaction as Conversation*, MoLIC¹, é uma linguagem criada para modelar a interação entre o usuário e o sistema como uma conversa entre o usuário e o projetista do sistema. A linguagem é uma ferramenta epistêmica criada pelo *Semiotic Engineering Research Group* [Sangiorgi e Barbosa 2009, Sangiorgi e Barbosa 2010].

2.2.2 Métodos de Prototipação de Interface

Este trabalho apresenta um método de modelagem focado na modelagem navegacional de um sistema. Para auxílio da aplicação do método foram utilizados métodos de protótipos de interface, descritos na Definição 2.14 e 2.15.

Definição 2.14. (*Sketch*)

Um *Sketch* é um esboço de um momento determinado da interface do sistema. Pode ser feito com desenhos à mão, ou utilizando alguma aplicação gráfica. O *Sketch* mostra a aparência de uma tela do sistema.

Algumas ferramentas foram criadas para auxiliar o desenvolvimento de *Sketches*. Neste trabalho é utilizada a ferramenta *Pencil*², da empresa *Evolus*, desenvolvido sob a plataforma Mozilla Firefox. **Notação.** Neste trabalho o *Sketch*, utilizado para representar as telas do sistema, é chamado de *Esboço de Tela*.

Definição 2.15. (*Storyboards*)

Um *Storyboard* é uma representação gráfica da aparência externa do sistema, sem o acompanhamento de funcionalidade. [Dix 2004]

A origem de *Storyboards* é a indústria de filmes, em que uma série de desenhos é usada para representar uma sequência de eventos relacionada com uma cena. Essa ideia é aplicada no projeto de interface, de forma que desenhos, muitas vezes feitos à mão, são utilizados para mostrar telas do sistema. No entanto, esses desenhos apenas mostram

¹<http://code.google.com/p/molic-designer/>

²<http://pencil.evolus.vn/en-US/Home.aspx>

superficialmente como o sistema será, não mostrando como o sistema funciona, pois não é possível acrescentar funcionalidades.

Um *Storyboard* de um sistema é uma sequência de imagens que representam um fluxo de execução de um evento, as imagens que formam um *storyboard* são *sketches* do sistema.

Capítulo 3

MNLAC: Modelagem Navegacional Baseada em Lógica Modal

Este trabalho propõe uma modelagem navegacional de um sistema de telas e o estudo de sua dinâmica, utilizando Lógica Temporal, chamado MNLAC, Modelagem Navegacional baseada em Lógica de Árvore Computacional. Este capítulo apresenta uma modelagem do sistema de telas utilizada na análise do sistema navegacional.

3.1 Elicitação de Requisitos

Na modelagem do sistema de telas, o MNLAC considera a elicitação de requisitos definida pelo modelo proposto em [Oliveira et al. 2010d, Oliveira 2011], denominado de *Árvore de Características*, como fundamento para a exemplificação da modelagem navegacional.

No modelo de *Árvores de Características*, a definição dos requisitos de um sistema, é baseado em uma metodologia de desenvolvimento guiado por características. Nesse contexto, uma característica é “uma função valorizada pelo cliente” que deve ser construída e acoplada ao sistema em construção.

Assim, o cliente deve indicar as características do sistema em linguagem natural e de forma gráfica, o que facilita a comunicação entre o cliente e o desenvolvedor. A Figura 3.1 mostra um exemplo de *Árvore de Características* que será utilizado ao longo desse trabalho como exemplo de aplicação.

A *Árvore de Característica* da Figura 3.1 [Oliveira et al. 2010d, Oliveira 2011] foi gerada com auxílio de 44 professores do ensino fundamental da rede municipal de Uberlândia. Cada professor, em separado, criou uma árvore de característica para uma comunidade virtual de aprendizagem, conforme descrito em [Oliveira et al. 2010d, Oliveira 2011]. Todos os itens apontados pelos professores foram considerados e unidos em uma

só árvore, pelo projetista do software.

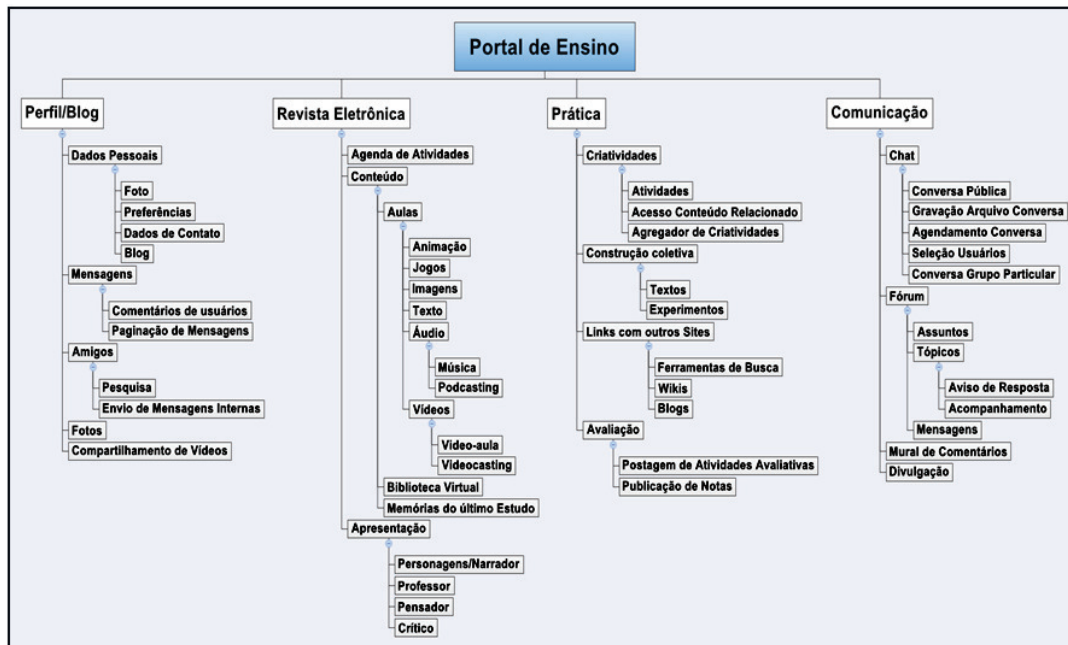


Figura 3.1: *Árvore de Características* de um portal educativo

Cada nó da árvore representa uma característica do sistema a ser desenvolvido. Nesse caso, uma característica foi definida como uma propriedade ou funcionalidade do sistema. A *Árvore de Características* foi utilizada para o processo de elicitação de requisitos, no momento de decisão de requisitos do projeto de *software*.

3.2 Geração do Grafo de Telas

Após definir as características necessárias para o desenvolvimento do sistema, a árvore foi então traduzida para um grafo de telas [Oliveira 2011].

Antes de realizar essa tradução, algumas estratégias foram consideradas:

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).
3. Retirada das folhas e nós da árvore de requisitos que representam requisitos não funcionais;
4. Geração do *Grafo de Telas*;

Após a obtenção da *Árvore de Características*, cada requisito funcional da árvore é transformado em uma tela. Nessa transformação, é considerada a forma como os itens serão dispostos na interface final e como eles serão interligados.

Por exemplo, o *Chat* na árvore da Figura 3.1 [Oliveira et al. 2010d, Oliveira 2011] possui os seguintes itens relacionados:

- *Conversa Pública*
- *Gravação Arquivo Conversa*
- *Agendamento Conversa*
- *Seleção Usuário*
- *Conversa Grupo Particular*

A *Conversa Pública* e *Seleção de Usuários* são funcionalidades incorporadas na tela *Chat*, ou seja, são funções inerentes da tela chat.

No entanto a *Gravação de Arquivo da Conversa*, o *Agendamento de Conversa* e *Conversa de Grupo Particular*, são funcionalidades que geram novas telas.

Dessa forma, cada item funcional identificado na *Árvore de Características* deve ser avaliado segundo a propriedade que ele possui de gerar uma nova tela.

Caso o item gere nova tela, então ele deve ser adicionado ao *Grafo de Telas* como um vértice. Caso contrário, o item é desconsiderado, sendo útil apenas no momento da criação dos esboços das telas.

A Figura 3.2 [Oliveira 2011] apresenta o *Grafo de Telas* final criado a partir da *Árvore de Características* da Figura 3.1 [Oliveira et al. 2010d, Oliveira 2011]. Observe que o *Grafo de Telas* permite visualizar as telas da interface do sistema e como elas se interconectam.

Essa visualização facilita o projeto da interface de usuário através do esboço das telas do sistema.

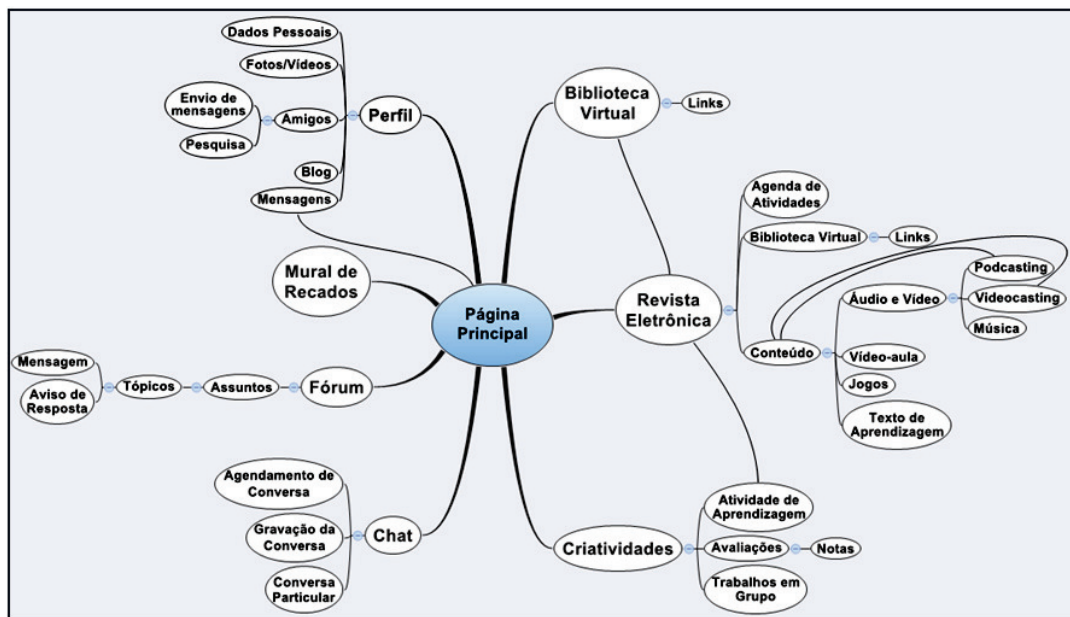


Figura 3.2: Grafo de Telas

3.3 Protótipo da Interface

Um esboço de tela da interface é uma ferramenta utilizada por projetistas para prototipação de interfaces com o usuário. Seu objetivo é possibilitar a visualização do sistema antes de começar a programá-lo.

O esboço da interface contém todos os elementos que o sistema teria no final do desenvolvimento. Ele pode ser criado com apoio do *Grafo de Telas* e da *Árvore de Características*. Isso porque o *Grafo de Telas* contém todas as funcionalidades do sistema que geram telas, dessa forma é possível se pautar nos vértices do grafo para criação dos esboços. Assim, para cada vértice do grafo está associado um esboço, ou seja, uma tela.

Além de ser utilizada para determinar o *Grafo de Telas*, a *Árvore de Característica* é, também, utilizada como auxílio para definição do conteúdo de cada esboço.

Definição 3.1. (Itens de Tela)

Um item de tela é um elemento inserido na composição de uma tela da interface do sistema. Um item de tela pode ser, por exemplo, um elemento como:

- Um Link,
- Uma Caixa de Seleção,
- Um Item de Menu,
- Um Botão.

Notação: Um item de tela é denotado, neste trabalho, pelo seu nome precedido pela letra *i*.

As denotações a seguir correspondem a itens de tela:

$$\begin{aligned} & iRevista, iForum, iPerfil, iMural, iBiblioteca, \\ & iCriatividade, iMensagens, iAgendamento, iGravar, \\ & iParticular, iFechar, iMinimizar \end{aligned}$$

Definição 3.2. (Funções Clicáveis e Não Clicáveis)

Os itens de tela são classificados em:

- Item clicável (Notação: α_1);
- Item não clicável (Notação: α_2).
- Os itens clicáveis são requisitos da **Árvore de Características** que representam, em uma tela, uma função ativa que dispara um evento. Os itens clicáveis são divididos em:
 - Item clicável que gera tela: Esses itens, ao sofrerem o clique do mouse, disparam um evento que cria uma nova tela. Um item desse tipo é denotado, neste trabalho, por α_{11} .
 - Item clicável que não gera tela: Esses itens, ao sofrerem o clique do mouse, disparam um evento que não cria tela. Isto é, ele dispara um evento interno de uma tela. Um item desse tipo é denotado, neste trabalho, por α_{12} .
- Um item não clicável é um item de tela que se encontra desativado. Ou seja, está presente na tela, mas ao clicar com o mouse nele, nenhum evento é disparado. Neste trabalho, um item desse tipo é denotado por α_2 .

Definição 3.3. (Tela)

Uma tela é um vértice do **Grafo de Telas** e corresponde a um mundo na semântica da Lógica Modal.

Uma tela é composta de itens clicáveis e não clicáveis, sendo denotada pela lista **Tela**, tal que:

$$Tela = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

Nessa notação,

- α_{11} é o conjunto de funções de tela clicáveis que geram novas telas
- α_{12} é o conjunto de funções de tela clicáveis que não geram telas
- α_2 é o conjunto de funções de tela não clicáveis.

Um item do tipo α_{11} é um item que ao ser clicado causa a exibição de outra tela. Em outras palavras, ao ser clicado o sistema se modifica, deslocando-se para outro vértice do **Grafo de Telas**, ou mundo possível, conforme a semântica da Lógica Modal.

Um item do tipo α_{12} não causa a mudança de vértice no grafo.

Notação. Uma tela é denotada, neste trabalho, pelo seu nome precedido pela letra t . As palavras a seguir denotam telas:

$$tChat, tAgendamento, tGravacao, tParticular$$

Exemplo 3.1. Tela $tChat$

Na Figura 3.3 é apresentado um esboço da tela $tChat$. Essa tela pode ser definida da seguinte forma:

$$tChat = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que:

$$\begin{aligned} a_{11} &= \{iRevista, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens, \\ &\quad iAgendamento, iGravar, iParticular, iFechar, iMinimizar\} \\ a_{12} &= \{iEnviar, iSelecionar, iMaximizar\} \\ a_2 &= \{iChat\} \end{aligned}$$

Observa-se pelo **Grafo de Telas** da Figura 3.2 que os itens

$$\{iRevista, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens, \\ iAgendamento, iGravar, iParticular, iFechar, iMinimizar\}$$

são links para outros vértices do **Grafo de Telas**. Os itens $iFechar$ e $iMinimizar$ são itens que causam a exibição de outra tela. Portanto são itens do tipo α_{11} .

Através do **Grafo de Telas**, observa-se que a tela $tChat$ possui conexão com as telas $tAgendamento$, $tGravação$ e $tParticular$. Essas telas são referenciadas no esboço da tela $tChat$, Figura 3.3, pelos botões $iAgendamento$, $iGravar$, $iParticular$, que ao serem clicados, ativam as respectivas telas.

No entanto, os itens

$$\{iEnviar, iSelecionar, iMaximizar\}$$

são funcionalidades necessárias que não estão presentes no *Grafo de Telas*, por não referenciarem telas, mas estão presentes na *Árvore de Características*, Figura 3.1, listados como requisitos funcionais. A funcionalidade *iMaximizar* é um item que não causa a exibição de outra tela. Dessa forma esses itens são do tipo α_{12} . Por fim, o item

$$\{iChat\}$$

é um item do tipo α_2 , pois está desativado. Ou seja, não executa nenhum evento ao ser clicado.

Tendo em vista que a tela *tChat* já está ativa, então, ao clicar no menu *Chat* nenhuma mudança no sistema de telas é realizada.

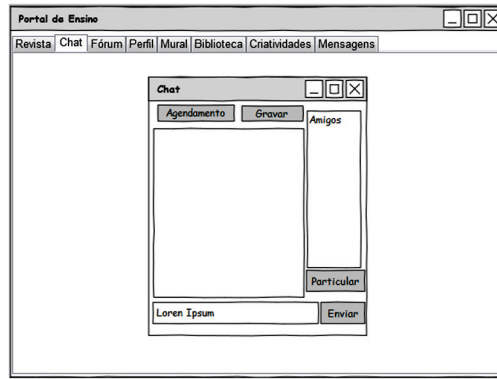


Figura 3.3: Tela de Chat

Exemplo 3.2. Tela Principal

Na Figura 3.4 apresenta-se um esboço da tela *tPrincipal*. Essa tela pode ser definida da seguinte forma:

$$tPrincipal = [\alpha_{11}[iRevista, iChat, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens], \alpha_{12}[\], \alpha_2[\]]$$

A tela Principal possui apenas itens do tipo α_{11} .

Exemplo 3.3. Tela Fórum

A Figura 3.5 apresenta um esboço da tela *tForum*, definida da seguinte forma:

$$tForum = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

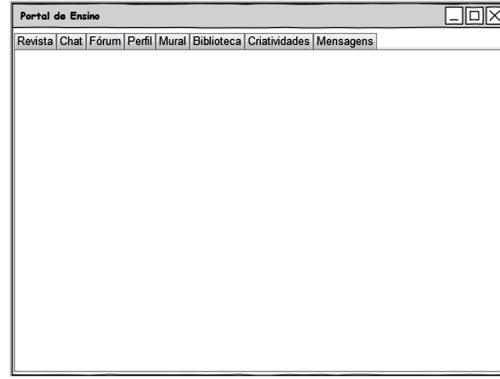


Figura 3.4: Tela Principal

tal que

$$a_{11} = \{iRevista, iChat, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens, iAssunto, iAddAssunto, iFechar, iMinimizar\}$$

$$a_{12} = \{iMaximizar\}$$

$$a_2 = \{iForum\}$$

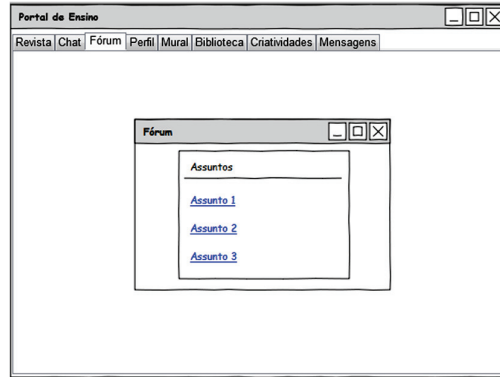


Figura 3.5: Tela Principal

Exemplo 3.4. Outras telas do Sistema

A seguir, descrevem-se as outras telas do **Grafo de Telas** da Figura 3.2:

$$tRevista = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$a_{11} = \{iChat, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens, iAgenda, iConteudo, iAtividades, iFechar, iMinimizar\}$$

$$a_{12} = \{iMaximizar\}$$

$$a_2 = \{iRevista\}$$

$$tPerfil = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iMural, iBiblioteca, iCriatividade, iMensagens, \\ &\quad iDadosPessoais, iEditarDados, iFotosVideos, iGerenciarFotosVideos, iAmigos, \\ &\quad iGerenciarAmigos, iBlog, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \\ a_2 &= \{iPerfil\} \end{aligned}$$

$$tBiblioteca = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iPerfil, iMural, iCriatividade, iMensagens, \\ &\quad iLinks, iAddLinks, iAddArquivos, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \\ a_2 &= \{iBiblioteca\} \end{aligned}$$

$$tCriatividades = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iPerfil, iMural, iBiblioteca, iMensagens, iAtividades, \\ &\quad iAddAtividades, iTrabalhos, iAddTrabalhos, iAvaliacoes, iAddAvaliacoes, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \\ a_2 &= \{iCriatividades\} \end{aligned}$$

$$tMural = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iPerfil, iCriatividades, iBiblioteca, iMensagens\} \\ a_{12} &= \{iEnviarRecado, iSelecionarRecados, iExcluirRecados, iMaximizar\} \\ a_2 &= \{iMural\} \end{aligned}$$

$$tAssuntos = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, \\ &\quad iTopicos, iAddTopico, iFechar, iMinimizar\} \end{aligned}$$

$$a_{12} = \{iMaximizar\}$$

$$a_2 = \{iForum\}$$

$$tTopicos = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$a_{11} = \{iRevista, iChat, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, iMsgs, iAddMsg, iAvisoResposta, iFechar, iMinimizar\}$$

$$a_{12} = \{iMaximizar\}$$

$$a_2 = \{iForum\}$$

$$tMsgs = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$a_{11} = \{iRevista, iChat, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, iEnviar, iCancelar, iFechar, iMinimizar\}$$

$$a_{12} = \{iMaximizar\}$$

$$a_2 = \{iForum\}$$

$$tAvisoResposta = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$a_{11} = \{iRevista, iChat, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, iConfirmarAviso, iCancelar, iFechar, iMinimizar\}$$

$$a_{12} = \{iMaximizar\}$$

$$a_2 = \{iForum\}$$

$$tAgendamento = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$a_{11} = \{iRevista, iForum, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, iSalvar, iCancelar, iFechar, iMinimizar\}$$

$$a_{12} = \{iData, iHora, iDescricao, iMaximizar\}$$

$$a_2 = \{iChat\}$$

$$tGravacao = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$a_{11} = \{iRevista, iForum, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, iSalvar, iCancelar, iFechar, iMinimizar\}$$

$$a_{12} = \{iSelLocal, iMaximizar\}$$

$$a_2 = \{iChat\}$$

$$tParticular = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$a_{11} = \{iRevista, iForum, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, iEnviar, iFechar, iMinimizar\}$$

$$a_{12} = \{iMaximizar\}$$

$$a_2 = \{iChat\}$$

$$tLinks = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$a_{11} = \{iRevista, iChat, iForum, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, iEditarLink, iExcluirLink, iFechar, iMinimizar\}$$

$$a_{12} = \{iMaximizar\}$$

$$a_2 = \{iBiblioteca\}$$

Exemplo 3.5. Telas Fórum e Chat

Considere um exemplo no qual o sistema está exibindo duas telas: $tChat$ e $tForum$. Considere, também, que a tela $tChat$ está ativa e a tela $tForum$ não está ativa. Nesse caso, a tela $tChat$ é definida como se segue:

$$tChat = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$a_{11} = \{iRevista, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens, iAgendamento, iGravar, iParticular, iFechar, iMinimizar\}$$

$$a_{12} = \{iEnviar, iSelecionarAmigos, iMaximizar\}$$

$$a_2 = \{iChat, iAssuntos, iAddAssunto, iMinimizar, iMaximizar, iFechar\}$$

Os itens

$$iAssuntos, iAddAssunto, \\ iMinimizar, iMaximizar, iFechar$$

são adicionados à definição da tela $tChat$, no conjunto α_2 . Isso, porque são itens que estão presentes na tela, mas não estão ativos. Observe que tais itens pertencem à tela $tForum$, a qual é representada por uma janela que está em segundo plano, como é mostrado na Figura 3.6.

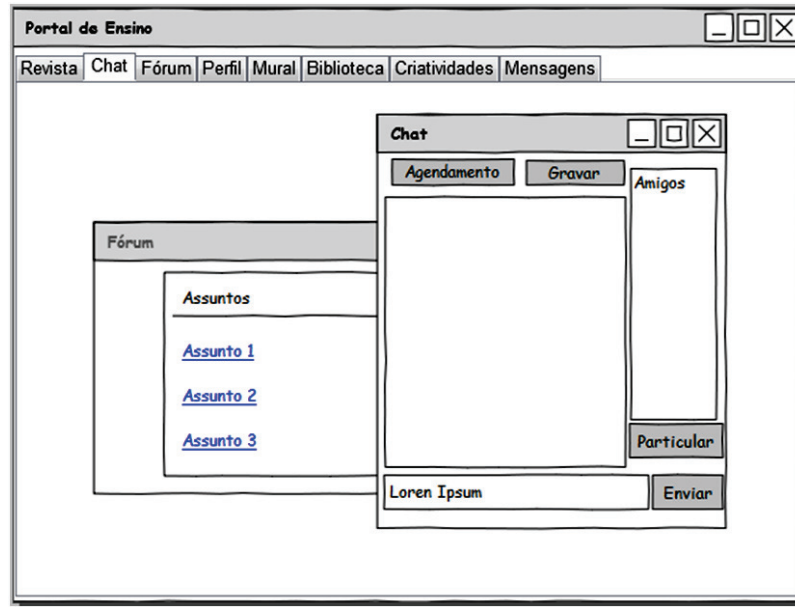


Figura 3.6: Tela Chat e Fórum

Definição 3.4. (Relação entre telas)

Duas telas t_1 e t_2 estão relacionadas por uma relação R se ao clicar em um item a do tipo α_{11} da tela t_1 , a tela t_2 é exibida. Ou seja,

$$\begin{aligned} (t_1, t_2) \in R \text{ se, e somente se,} \\ t_1 = [\alpha_{11}, \alpha_{12}, \alpha_2] \text{ tal que} \\ a \in \alpha_{11}, e \\ a \text{ é um link para tela } t_2 \end{aligned}$$

Uma relação é representada no **Grafo de Telas** como um arco que liga as telas.

Exemplo 3.6. Relação entre $tPrincipal$ e $tChat$

No **Grafo de Telas** da Figura 3.2, as telas

$$tPrincipal \text{ e } tChat$$

estão relacionadas por arcos. Ou seja,

$$(tPrincipal, tChat) \in R.$$

Isso ocorre porque existe um item do tipo α_{11} que pertence à tela $tPrincipal$ que ao ser clicado causa a execução da funcionalidade que gera a tela $tChat$. Neste exemplo, o

$$iChat,$$

representado pelo botão de menu **Chat** da Figura 3.4, representa o *link* entre a tela **Principal** e a tela **Chat**.

Definição 3.5. (*Frame* de Telas)

Todo **Grafo de Telas**, com as suas relações, corresponde ao Frame da Lógica Modal. Esse Frame é denominado Frame de Telas.

Exemplo 3.7. *Frame* de Telas

Uma tela do tipo

$$t_1 = [\alpha_{11}[a, b, c], \alpha_{12}[d, e], \alpha_2[f]]$$

corresponde, no contexto da Lógica Modal, a um mundo possível do Frame.

Definição 3.6. (Modelo Navegacional)

Um Frame de Telas, juntamente com uma função de valoração aplicada ao Frame constitui um **Modelo Navegacional**.

A função de valoração, inicialmente definida no contexto da Lógica Modal, corresponde no modelo proposto aos conjuntos de itens: α_{11} e α_{12} pertencentes à Tela.

Portanto, um **Modelo Navegacional** é um frame, tal que a cada nó w do **Grafo de Telas** é aplicada a função de valoração do tipo:

$$V(w) = \alpha_{11}, \alpha_{12}$$

Um Modelo pode ser representado na forma da Figura 3.7. Nesta figura é apresentada uma parte do **Modelo Navegacional** do sistema dado pelo **Grafo de Telas** da Figura 3.2.

Cada definição de tela é esquematizada na forma de um vértice do Frame, com o nome da tela, e os seus itens dos tipos α_{11} e α_{12} .

3.4 Modelo Navegacional baseado em Lógica de Árvore Computacional

No MNLAC, o conjunto de **Esboços de Telas** são convertidos para o **Modelo Navegacional**.

O **Modelo Navegacional** representa as Telas do Sistema e suas relações.

Assim, cada funcionalidade clicável do esboço é colocada no modelo como um item de tela.

Exemplo 3.8. Vértice $tPrincipal$

A Tela Principal, por exemplo, definida no Exemplo 3.2, possui os seguintes itens dos tipos α_{11} e α_{12} :

$iRevista, iChat, iPerfil, iMural,$
 $iBiblioteca, iCriatividades, iMensagens.$

Esses itens são representados no **Modelo Navegacional** da Figura 3.7.

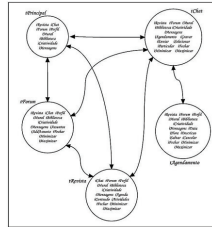


Figura 3.7: **Modelo Navegacional**

A partir da definição dos vértices do modelo navegacional, a proposta apresentada nesse trabalho descreve tal modelo, utilizando, também, a linguagem da Lógica Modal Temporal.

Nesse sentido, o **Modelo Navegacional** é representado como um par

$$\mathcal{M} = (\mathfrak{F}, V)$$

onde $\mathfrak{F}$ é um frame e V é uma função de valoração que atribui a cada mundo w , ou tela, em W , um subconjunto $V(w)$ de símbolos proposicionais em ϕ .

Isto é, $V(w)$ é um conjunto de itens do tipo α_{11} e α_{12} , que são verdadeiros no mundo, ou tela, w .

Um *frame*, como o **Grafo de Telas**, é uma estrutura relacional simples com relações binárias.

Nesse caso, a estrutura relacional é uma tupla

$$\mathfrak{F} = (W, R)$$

O primeiro elemento da tupla é um conjunto não vazio W , chamado de universo, ou domínio de $\mathfrak{F}$. O segundo elemento da tupla é o conjunto R de relações em W [Blackburn et al. 2002].

No MNLAC, os elementos de W são as telas do sistema que correspondem aos mundos possíveis da Lógica Modal.

A Figura 3.7 apresenta uma estrutura relacional $\mathfrak{F} = \{W, R\}$, na qual

$$W = \{tPrincipal, tChat, tRevista, tForum\},$$

e

$$\begin{aligned} (w_1, w_2) \in R \text{ se, e somente se,} \\ w_1 = [\alpha_{11}, \alpha_{12}, \alpha_2] \text{ tal que} \\ a \in \alpha_{11}, \text{ e } a \text{ é um link para tela } w_2 \end{aligned}$$

Além disso, no caso da Figura 3.7, o conjunto ϕ das proposições do modelo é dado por:

$$\begin{aligned} \phi = \{ & iRevista, iChat, iForum, iPerfil, iMural, \\ & iBiblioteca, iCriatividade, iMensagens, iAgendamento, \\ & iGravar, iEnviar, iSelecionar, iParticular, \\ & iFechar, iMinimizar, iMaximizar, iAssuntos, \\ & iAddAssunto, iData, iHora, iDescricao, \\ & iSalvar, iCancelar, iAgenda, iConteudo, iAtividades \} \end{aligned}$$

Exemplo 3.9. Conjunto Universo do Modelo

Cada nó do grafo de telas é expandido, considerando as funcionalidades dos tipos α_{11} e α_{12} correspondentes. Assim, o **Grafo de Telas** da Figura 3.2 pode ser redesenhado de forma que cada vértice se torne um conjunto de itens.

Os vértices que compõem o conjunto universo W de $\mathcal{M}$, com suas funções de valoração V , são os seguintes conjuntos:

$$tPrincipal = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens\} \\ a_{12} &= \{ \} \end{aligned}$$

$$tChat = [\alpha_1 1, \alpha_1 2]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens, \\ & iAgendamento, iGravar, iParticular, iFechar, iMinimizar\} \end{aligned}$$

$$a_{12} = \{iEnviar, iSelecionar, iMaximizar\}$$

$$tForum = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens, \\ &\quad iAssuntos, iAddAssunto, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

$$tRevista = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iChat, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens, \\ &\quad iAgenda, iConteudo, iAtividades, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

$$tPerfil = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iMural, iBiblioteca, iCriatividade, iMensagens, \\ &\quad iDadosPessoais, iEditarDados, iFotosVideos, iGerenciarFotosVideos, iAmigos, \\ &\quad iGerenciarAmigos, iBlog, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

$$tBiblioteca = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iPerfil, iMural, iCriatividade, iMensagens, \\ &\quad iLinks, iAddLinks, iAddArquivos, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

$$tCriatividades = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iPerfil, iMural, iBiblioteca, iMensagens, iAtividades, \\ &\quad iAddAtividades, iTrabalhos, iAddTrabalhos, iAvaliacoes, iAddAvaliacoes, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

$$tMural = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iPerfil, iCriatividades, iBiblioteca, iMensagens\} \\ a_{12} &= \{iEnviarRecado, iSelecionarRecados, iExcluirRecados, iMaximizar\} \end{aligned}$$

$$tAssuntos = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iPerfil, iMural, iCriatividades, iBiblioteca, \\ &\quad iMensagens, iTopicos, iAddTopico, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

$$tTopicos = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, \\ &\quad iMsgs, iAddMsg, iAvisoResposta, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

$$tMsgs = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, \\ &\quad iEnviar, iCancelar, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

$$tAvisoResposta = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, \\ &\quad iConfirmarAviso, iCancelar, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

$$tAgendamento = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iForum, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, \\ &\quad iSalvar, iCancelar, iFechar, iMinimizar\} \\ a_{12} &= \{iData, iHora, iDescricao, iMaximizar\} \end{aligned}$$

$$tGravacao = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iForum, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, \\ &\quad iSalvar, iCancelar, iFechar, iMinimizar\} \\ a_{12} &= \{iSelLocal, iMaximizar\} \end{aligned}$$

$$tParticular = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iForum, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, \\ &\quad iEnviar, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

$$tLinks = [\alpha_{11}, \alpha_{12}]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iPerfil, iMural, iCriatividades, iBiblioteca, iMensagens, \\ &\quad iEditarLink, iExcluirLink, iFechar, iMinimizar\} \\ a_{12} &= \{iMaximizar\} \end{aligned}$$

Exemplo 3.10. Conjunto de relações no Modelo

O Conjunto de relações R em W do **Modelo Navegacional**, determinado a partir do **Grafo de Telas** da Figura 3.2, considerando as telas definidas no Exemplo 3.7 e o **Modelo Navegacional** da Figura 3.7, é dado por:

$$\begin{aligned} R = \{ & (tPrincipal, tPerfil), (tPerfil, tPrincipal), \\ & (tPrincipal, tRevista), (tRevista, tPrincipal), \\ & (tPrincipal, tBiblioteca), (tBiblioteca, tPrincipal), \\ & (tPrincipal, tCriatividades), (tCriatividades, tPrincipal), \\ & (tPrincipal, tChat), (tChat, tPrincipal), \\ & (tPrincipal, tForum), (tForum, tPrincipal), \\ & (tPrincipal, tMural), (tMural, tPrincipal), \\ & (tPrincipal, tMensagens), (tMensagens, tPrincipal), \\ & (tPerfil, tMensagens), (tMensagens, tPerfil), \\ & (tRevista, tBiblioteca), (tBiblioteca, tRevista), \\ & (tBiblioteca, tLinks), (tLinks, tBiblioteca), \\ & (tChat, tAgendamento), (tAgendamento, tChat), \\ & (tChat, tGravacao), (tGravacao, tChat), \end{aligned}$$

$$\begin{aligned}
& (tChat, tParticular), (tParticular, tChat), \\
& (tForum, tAssuntos), (tAssuntos, tForum), \\
& (tAssuntos, tTopicos), (tTopicos, tAssuntos), \\
& (tTopicos, tMsgs), (tMsgs, tTopicos), \\
& (tTopicos, tAvisoResposta), (tAvisoResposta, tTopicos), \\
& (tChat, tRevista), (tRevista, tChat), \\
& (tChat, tForum), (tForum, tChat) \dots \}
\end{aligned}$$

Em seguida, no **Modelo Navegacional**, a partir da semântica da Lógica Modal, testa-se a validade de fórmulas da Lógica Modal. Considere os exemplos a seguir,

Exemplo 3.11. Fórmula do tipo p

Considerando a fórmula

$$iRevista$$

na tela

$$tPrincipal$$

Considerando a dedução:

$$\mathcal{M}, tPrincipal \Vdash iRevista$$

Se a tela ativa do sistema é a tela $tPrincipal$, então a dedução acima é satisfeita se:

$$iRevista \in V(tPrincipal)$$

e

$$iRevista \in \phi$$

Observa-se que, conforme a Figura 3.7, as duas condições acima são satisfeitas. Logo,

$$\mathcal{M}, tPrincipal \Vdash iRevista$$

Exemplo 3.12. Fórmula do tipo $\neg p$

Considerando a fórmula

$$\neg iAgendamento$$

na tela

$$tPrincipal$$

Considerando a dedução:

$$\mathcal{M}, tPrincipal \Vdash \neg iAgendamento$$

Ou seja,

$$\mathcal{M}, tPrincipal \nVdash iAgendamento$$

Se a tela ativa do sistema é a tela $tPrincipal$, então a dedução acima é satisfeita se:

$$iAgendamento \notin V(tPrincipal)$$

e

$$iAgendamento \in \phi$$

Observa-se que, conforme a Figura 3.7, as duas condições acima são satisfeitas. Logo,

$$\mathcal{M}, tPrincipal \Vdash \neg iAgendamento$$

Exemplo 3.13. Fórmula do tipo $p \vee q$

Considerando a fórmula

$$iConteudo \vee iAtividades$$

na tela

$$tRevista$$

Considerando a dedução:

$$\mathcal{M}, tRevista \Vdash iConteudo \vee iAtividades$$

Ou, equivalentemente,

$$\mathcal{M}, tRevista \Vdash iConteudo$$

ou

$$\mathcal{M}, tRevista \Vdash iAtividades$$

Se a tela ativa do sistema é a tela $tRevista$, então a dedução acima é satisfeita se:

$$iConteudo \in V(tRevista)$$

ou

$$iAtividades \in V(tRevista)$$

e

$$iConteudo \text{ e } iAtividades \in \phi$$

Observa-se que, no caso da Figura 3.7, as três condições acima são satisfeitas. Logo,

$$\mathcal{M}, tRevista \Vdash iConteudo \vee iAtividades$$

Exemplo 3.14. Fórmula do tipo $p \wedge q$

Considerando a fórmula

$$iChat \wedge iForum$$

na tela

$$tForum$$

Considere a dedução:

$$\mathcal{M}, tForum \Vdash iChat \wedge iForum$$

Ou seja,

$$\mathcal{M}, tForum \Vdash iChat$$

e

$$\mathcal{M}, tForum \Vdash iForum$$

Se a tela ativa do sistema é a tela $tForum$, então a dedução acima é satisfeita se:

$$iChat \in V(tForum)$$

e

$$iForum \in V(tForum)$$

e

$$iChat \text{ e } iForum \in \phi$$

Observa-se que, no caso da Figura 3.7, a condição:

$$iForum \in V(tForum)$$

não é satisfeita. Logo,

$$\mathcal{M}, tForum \not\models iChat \wedge iForum$$

Exemplo 3.15. Fórmula do tipo $\Diamond p$

Considerando a fórmula

$$\Diamond iBiblioteca$$

na tela

$$tPrincipal$$

Considerando a dedução:

$$\mathcal{M}, tPrincipal \models \Diamond iBiblioteca$$

Se a tela ativa do sistema é a tela $tPrincipal$, então a dedução acima é satisfeita se:

$$\exists w \in W, \text{tal que } (tPrincipal, w) \in R$$

e

$$iBiblioteca \in V(w)$$

e

$$iBiblioteca \in \phi$$

Observa-se que, no caso da Figura 3.7, a condição:

$$\exists w \text{ tal que } (tPrincipal, w) \in R$$

é satisfeita para a tela $tRevista$. Ou seja,

$$(tPrincipal, tRevista) \in R$$

e

$$\mathcal{M}, tRevista \Vdash iBiblioteca$$

Logo:

$$\mathcal{M}, tPrincipal \Vdash \Diamond iBiblioteca$$

Exemplo 3.16. Fórmula do tipo $\Box p$

Considerando a fórmula

$$\Box iFechar$$

na tela

$$tPrincipal$$

Considerando a dedução:

$$\mathcal{M}, tPrincipal \Vdash \Box iFechar$$

Se a tela ativa do sistema é a tela $tPrincipal$, então a dedução acima é satisfeita se:

$$\forall w \in W, \text{tal que } (tPrincipal, w) \in R$$

e

$$iFechar \in V(w)$$

e

$$iFechar \in \phi$$

Observa-se que, no caso da Figura 3.7, a condição:

$$\forall w \in W, \text{tal que } (tPrincipal, w) \in R$$

é satisfeita para todas as telas que possuem relação com a tela $tPrincipal$. Ou seja,

$$(tPrincipal, tRevista), (tPrincipal, tChat), (tPrincipal, tForum) \in R$$

e

$$\mathcal{M}, tRevista \models iFechar$$

e

$$\mathcal{M}, tForum \models iFechar$$

e

$$\mathcal{M}, tChat \models iFechar$$

Logo:

$$\mathcal{M}, tPrincipal \models \Box iFechar$$

De forma análoga, o **Modelo Navegacional** da Figura 3.7, também pode ser utilizado, considerando a Lógica Temporal Computacional, LAC.

Nos exemplos a seguir, analisa-se o uso de cada operador da LAC no **Modelo Navegacional** da Figura 3.7, e nas outras telas do modelo definidas no Exemplo 3.7, considerando as relações definidas no Exemplo 3.8.

Exemplo 3.17. Fórmula do tipo $AX(p)$

Considerando a fórmula

$$AX(iFechar)$$

Supondo que a tela ativa no sistema seja:

$$tPrincipal$$

Considerando a dedução:

$$\mathcal{M}, tPrincipal \models AX(iFechar)$$

A semântica da fórmula $AX(iFechar)$ é similar à da fórmula $\Box iFechar$ da Lógica Modal. Ou seja, a dedução semântica acima é verdadeira se:

$$\forall \pi^2 \subset \pi, \text{ tal que } tPrincipal \text{ é a primeira tela do caminho } \pi$$

e

$$iFechar \in V(w), \text{ tal que } w \text{ é a primeira tela do caminho } \pi^2$$

e

$$iFechar \in \phi$$

Observa-se que, no caso da Figura 3.7, essas condições são satisfeitas para todos os caminhos do modelo iniciados em $tPrincipal$.

Exemplo 3.18. Fórmula do tipo $EX(p)$

Considerando a fórmula

$$\neg EX(iChat \wedge iAgendamento)$$

Supondo que a tela ativa no sistema seja:

$$tPrincipal$$

Considerando a dedução:

$$\mathcal{M}, tPrincipal \Vdash \neg EX(iChat \wedge iAgendamento)$$

A semântica da fórmula $EX(iChat \wedge iAgendamento)$ é similar à da fórmula $\Diamond(iChat \wedge iAgendamento)$ da Lógica Modal. Ou seja, a dedução semântica acima é verdadeira se:

$$\forall \pi^2 \subset \pi, \text{ tal que } tPrincipal \text{ é a primeira tela do caminho } \pi$$

e

$$iChat \notin V(w), \text{ tal que } w \text{ é a primeira tela do caminho } \pi^2$$

ou

$$iAgendamento \notin V(w), \text{ tal que } w \text{ é a primeira tela do caminho } \pi^2$$

$$iChat, iAgendamento \in \phi$$

Observa-se que, no caso da Figura 3.7, essas condições são satisfeitas para todos os caminhos do modelo iniciados em $tPrincipal$. Ou seja, em nenhum caminho iniciado em $tPrincipal$, a segunda tela possui $iChat$ e $iAgendamento$ ao mesmo tempo.

Essa propriedade significa que só é possível fazer um **Agendamento de Conversa**, se a tela de **Chat** estiver ativa. O que de fato deve ser verdade no sistema, pois agendar uma conversa é uma função da tela **Chat**.

Exemplo 3.19. Fórmula do tipo $EG(p)$

Considerando a fórmula

$$EG(\neg iChat)$$

Supondo que a tela ativa no sistema seja:

$$tChat$$

Considerando a dedução:

$$\mathcal{M}, tChat \Vdash EG(\neg iChat)$$

A dedução semântica acima é verdadeira se:

$$\exists \pi, \text{tal que } tChat \text{ é a primeira tela do caminho}$$

e

$$\forall \pi^i \subset \pi, \text{com } i > 1$$

e

$$iChat \notin V(w), \text{tal que } w \text{ é a primeira tela do caminho } \pi^i$$

$$iChat \in \phi$$

Essas condições são satisfeitas para todas as telas de pelo menos um caminho do modelo iniciado em $tChat$. Ou seja, no caminho iniciado em $tChat$, seguindo pela tela $tAgendamento$, depois pela tela $tChat \dots$, nenhuma das telas possui o item $iChat$.

Ou seja,

$$tChat \rightarrow tAgendamento \rightarrow tChat \rightarrow tGravacao \dots$$

Essa propriedade significa que dentro do contexto da tela $Chat$, o botão $Chat$ não está ativo, ou seja, é um item do tipo α_2 .

Exemplo 3.20. Fórmula do tipo $AG(p)$

Considerando a fórmula

$$AG(iFechar)$$

Supondo que a tela ativa no sistema seja:

$$tPrincipal$$

Considerando a dedução:

$$\mathcal{M}, tChat \Vdash AG(iFechar)$$

A dedução semântica acima é verdadeira se:

$$\forall \pi, \text{tal que } tPrincipal \text{ é a primeira tela do caminho}$$

e

$$\forall \pi^i \subset \pi, \text{com } i > 1$$

e

$$iFechar \in V(w), \text{tal que } w \text{ é a primeira tela do caminho } \pi^i \\ iChat \in \phi$$

Essas condições são satisfeitas para todas as telas de todos os caminhos do modelo iniciado em *tPrincipal*.

Essa propriedade significa que em qualquer tela do sistema, existe o botão ***Fechar***.

Exemplo 3.21. Fórmula do tipo EF(p)

Considerando a fórmula

$$EF(iGravar)$$

Supondo que a tela ativa no sistema seja:

$$tPrincipal$$

Considerando a dedução:

$$\mathcal{M}, tPrincipal \Vdash EF(iGravar)$$

A dedução semântica acima é verdadeira se:

$$\exists \pi, \text{tal que } tPrincipal \text{ é a primeira tela do caminho}$$

e

$$\exists \pi^i \subset \pi$$

e

$iGravar \in V(w)$, tal que w é a primeira tela do caminho π^i e $iGravar \in \phi$

Essas condições são satisfeitas, pois existe um caminho iniciado em $tPrincipal$, em que em dado momento a opção de **Gravar Conversa** fica disponível.

Ou seja,

$$tPrincipal \rightarrow tChat$$

Essa propriedade permite descobrir se alguma funcionalidade do sistema será alcançável.

A MNLAC, proposta neste trabalho, é uma formalização, utilizando Lógica Modal, do fluxo das interfaces. Ou seja, o modelo possibilita a análise da navegação entre as diferentes telas de um sistema.

Nesse contexto, cada tela do modelo representa uma visão da interface do sistema em um determinado momento. Além disso, no modelo proposto é possível verificar a alcançabilidade de telas de um sistema. Isto é, que propriedades devem ser satisfeitas para, partindo de uma tela, ser possível alcançar outras telas (situações).

Portanto, além de modelar o fluxo das telas, o método também permite verificar propriedades que determinam o fluxo navegacional do sistema. A seguir são apresentados alguns exemplos de propriedades que uma interface usável deve possuir.

Exemplo 3.22. Atalhos

Uma interface deve prover acesso direto a partes importantes de um sistema. Por exemplo, considerando que a tela de **Perfil do Usuário** seja uma área que deve ter rápido acesso, em todas as páginas do sistema deve existir um link para acessá-la.

Essa propriedade poderia ser representada pela seguinte dedução:

$$\mathcal{M} \Vdash AG(iPerfil)$$

Ou seja, em qualquer caminho que o usuário seguir, partindo de qualquer tela, o item **Perfil** é clicável, permitindo, assim, o acesso direto à tela **Perfil**. O item $iPerfil$ pertence ao conjunto α_{11} de todas as telas do sistema.

No entanto, no modelo utilizado como exemplo neste trabalho, essa dedução é falsa. O item *iPerfil* não pertence à tela *tPerfil*, que é perfeitamente natural, pois a tela que se deseja atingir já está ativa. Dessa forma, deve-se reescrever a dedução da seguinte forma:

$$\mathcal{M} \models AG(iPerfil) \vee (tPerfil \rightarrow \neg iPerfil)$$

Essa dedução significa que, em todos os caminhos do modelo, na tela ativa o item **Perfil** é clicável ou se a tela ativa é a tela *tPerfil*, então o item *iPerfil* não é clicável, ou seja, pertence ao conjunto α_2 .

Exemplo 3.23. Ajuda

Para um sistema ser considerado usável, os usuários devem ter acesso à ajuda ou documentação de cada função do sistema. Uma das heurísticas de Nielsen para um sistema usável diz que é necessário prover *help* e documentação e essas informações devem ser fáceis de encontrar [Nielsen 1993].

Assim, é necessário testar o sistema para verificar se todas as telas possuem acesso à documentação. Nesse caso a dedução a ser feita é:

$$\mathcal{M} \models AG(ihelp)$$

No sistema de exemplo deste trabalho, a dedução $\mathcal{M} \models AG(ihelp)$ não é satisfeita. Não foi previsto na **Árvore de Característica** da Figura 3.1, nem no **Grafo de Telas** da Figura 3.2 o item *help*. Logo o **Modelo Navegacional** gerado não satisfaz a dedução.

O sistema deve, então, ser modificado para atender a esse requisito. A primeira alteração a ser feita é na **Árvore de Característica**, de forma a adequá-la. A Figura 3.8 apresenta a **Árvore de Característica** modificada. A característica *help* foi adicionada como um nó da Árvore.

O item **Help** é um requisito funcional da Árvore que deve gerar tela. Em qualquer página do sistema que estiver ativa, deve existir um link para uma tela de ajuda com conteúdo específico. Assim, deve-se representar o item **Help** como um vértice do Grafo de Telas que deve ter relação com todos os outros vértices do grafo.

A Figura 3.9 apresenta, então, o **Grafo de Telas** modificado, com a adição do vértice **Help** e suas relações.

Observação: O vértice **Help** e suas relações deveriam ter sido acrescentados com o mesmo estilo dos demais vértices, assim como suas relações. O vértice foi colocado em azul, e suas relações em linhas pontilhadas azuis, apenas para destacá-lo dos demais.

A última modificação a ser feita é no **Modelo Navegacional**. Uma tela foi adicionada, e consequentemente links para essa tela devem ser adicionados em todas as demais telas do sistema.

$$iBiblioteca, iMensagens, iHelp\}$$

Nesse caso, está sendo tratada a tela de **Help** chamada a partir da tela **Principal**. Foi decisão do projetista que a tela **Help** fique sobreposta a todas as outras, desativando, então, todos os links da tela chamadora. Assim, as únicas funções ativas para usuário são de fechar a janela, minimizá-la ou maximizá-la.

As demais telas do sistema devem ser alteradas, de forma a conter o link para a tela **Help**. Por exemplo, as telas **Principal**, **Revista** e **Perfil** ficam da seguinte forma:

$$tPrincipal = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, \\ &\quad iMensagens, iHelp\} \\ a_{12} &= \{ \} \\ a_2 &= \{ \} \end{aligned}$$

$$tRevista = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$\begin{aligned} a_{11} &= \{iChat, iForum, iPerfil, iMural, iBiblioteca, iCriatividade, iMensagens, \\ &\quad iAgenda, iConteudo, iAtividades, iFechar, iMinimizar, iHelp\} \\ a_{12} &= \{iMaximizar\} \\ a_2 &= \{iRevista\} \end{aligned}$$

$$tPerfil = [\alpha_{11}, \alpha_{12}, \alpha_2]$$

tal que

$$\begin{aligned} a_{11} &= \{iRevista, iChat, iForum, iMural, iBiblioteca, iCriatividade, iMensagens, \\ &\quad iDadosPessoais, iEditarDados, iFotosVideos, iGerenciarFotosVideos, iAmigos, \\ &\quad iGerenciarAmigos, iBlog, iFechar, iMinimizar, iHelp\} \\ a_{12} &= \{iMaximizar\} \\ a_2 &= \{iPerfil\} \end{aligned}$$

Nesse momento, a dedução

$$M \Vdash AG(iHelp)$$

ainda não é satisfeita, pois apesar de ser verdadeira em quase todas as telas em todos os caminhos que o usuário seguir, ela não é verdadeira quando a tela **tHelp** estiver ativa, pois na tela **tHelp** o item **iHelp** não é clicável. De fato, não faz sentido ter uma tela de

ajuda para a tela de ajuda. Assim, a fórmula

$$AG(iHelp)$$

deve, então, ser alterada para permitir essa exceção. A fórmula será, portanto:

$$AG(iHelp) \vee (tHelp \rightarrow \neg iHelp)$$

E a dedução

$$M \models AG(iHelp) \vee (tHelp \rightarrow \neg iHelp)$$

é satisfeita no modelo modificado. Pois em todos os caminhos, iniciados em qualquer tela, a fórmula

$$AG(iHelp) \vee (tHelp \rightarrow \neg iHelp)$$

será verdadeira.

Capítulo 4

Trabalhos Relacionados

No Capítulo 2 foram apresentados alguns métodos utilizados para especificação e modelagem de interfaces com usuário. Neste Capítulo são aprofundados os conceitos envolvidos nos métodos de modelagem OOHDM e MoLIC, a fim de comparativo com o método MNLAC, proposto neste trabalho.

4.1 Object Oriented Hypermedia Design Method

O método OOHDM é um método utilizado para auxiliar no desenvolvimento de aplicações hipermídia, e divide o processo de engenharia de desenvolvimento em quatro etapas [Schwabe e Rossi 1995, Henrichs 2005]:

- Modelagem Conceitual - Nesta fase é construído um esquema conceitual que representa objetos e relacionamentos existentes no domínio da aplicação. O esquema conceitual é um diagrama constituído de classes, relações e subsistemas.
- Modelagem Navegacional - Nesta fase podem ser construídos vários modelos navegacionais para um mesmo domínio, levando em consideração os perfis dos futuros usuários.
- Modelagem da Interface Abstrata - Nesta fase são definidos como estarão perceptíveis os objetos de interface, suas propriedades e transformações, não se preocupando com qual será o ambiente e a forma de implementação da aplicação.
- Implementação - Nesta fase é definido o ambiente de implementação, e a forma pela qual será realizado o mapeamento dos objetos navegacionais e de interface para o ambiente escolhido.

A Figura 4.1 [Schwabe e Rossi 1995, Henrichs 2005] esquematiza o processo de modelagem de software utilizando o OOHDM.

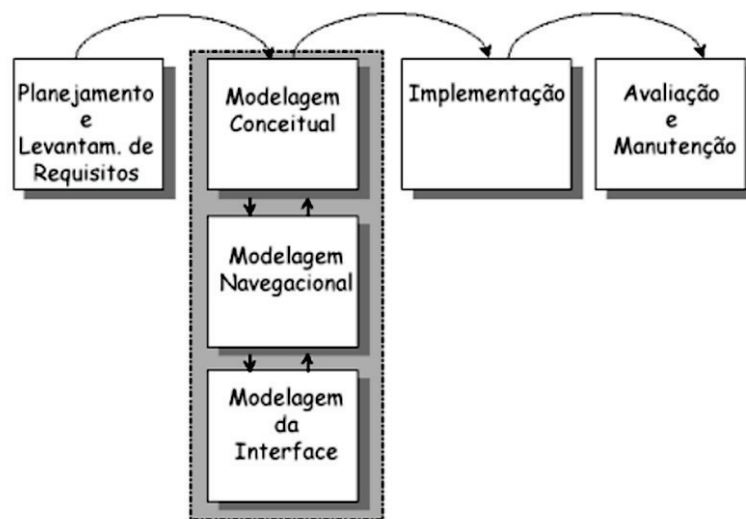


Figura 4.1: Modelo Proposto pelo OOHDM

O modelo proposto por este trabalho, MNLAC, se propõe a auxiliar na modelagem navegacional de uma interface com o usuário. O MNLAC possui o mesmo foco que a etapa de Modelagem Navegacional do OOHDM. Assim, este trabalho apresentará mais detalhadamente a etapa de Modelagem navegacional do OOHDM, a cunho de comparação com o método MNLAC.

4.1.1 Modelagem Navegacional

O OOHDM define a modelagem navegacional como visões do modelo conceitual [Schwabe et al. 1996]. Diferentes modelos navegacionais podem ser construídos para o mesmo esquema conceitual, cada um considerando conjuntos diferentes de usuários e tarefas.

“Uma aplicação em OOHDM é definida como visões navegacionais do modelo conceitual. Cada uma destas visões define um conjunto de contextos e classes navegacionais. Os contextos navegacionais expressam a estrutura navegacional geral da aplicação hipermídia, enquanto as classes navegacionais especificam os objetos que serão vistos pelo usuário” [Rossi 1996].

O projeto navegacional é subdividido em dois esquemas:

- Esquema de Classes Navegacionais - Conjunto de nós, elos e estruturas de acesso representados em um diagrama, que representa uma visão navegacional da aplicação. Este esquema define os relacionamentos entre os objetos navegacionais.

- Esquema de Contextos Navegacionais - Tem por objetivo definir em quais contextos será permitida a navegação entre as informações e quais informações serão apresentadas ao usuário [Vilain 2002]. Este esquema define a navegação em geral.

O nó de um modelo navegacional é um conjunto de instâncias que possuem características diferentes, ou seja, um nó é uma classe navegacional. Nós possuem atributos que representam as propriedades de suas instâncias, e operações, que são as reações dos nós quando são ativados durante a navegação. Os nós armazenam informações importantes para a realização de determinadas tarefas. A Figura 4.2 [Rossi 1996] apresenta um exemplo de um nó Pessoa e seus atributos.

Elos são a realização navegacional das relações. A Figura 4.3 [Rossi 1996] apresenta um exemplo de um elo entre o nó Obra e o nó Pessoa.

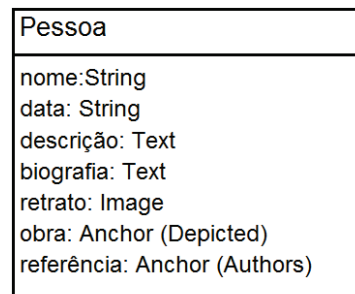


Figura 4.2: Exemplo de Nó.

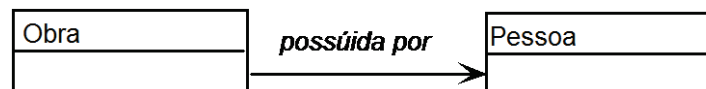


Figura 4.3: Exemplo de Elo.

Atributos específicos de cada nó, chamados de âncoras ou índices, representam, em conjunto com os elos, a navegação. As âncoras e índices são tratados como gatilhos navegacionais, ou seja, quando uma âncora ou índice for disparado uma navegação ocorre [Henrichs 2005].

A elaboração de contextos de navegação tem por objetivo definir em quais contextos será permitida a navegação entre as informações e quais informações serão apresentadas ao usuário [Vilain 2002].

Para obter acesso aos contextos de navegação é necessário especificar as estruturas de acesso. Uma estrutura de acesso é um conjunto de itens ordenados que funcionam como índices e auxiliam o usuário a encontrar a informação desejada [Schwabe 2003].

Os contextos de navegação determinam o conjunto de objetos de navegação acessíveis ao usuário em cada momento da aplicação [Henrichs 2005].

A Figura 4.4 [Rossi 1996] apresenta um esquema navegacional simples. A partir desse esquema pode-se verificar que é possível navegar de pinturas a pintores, e de cada um destes a seus contextos históricos.

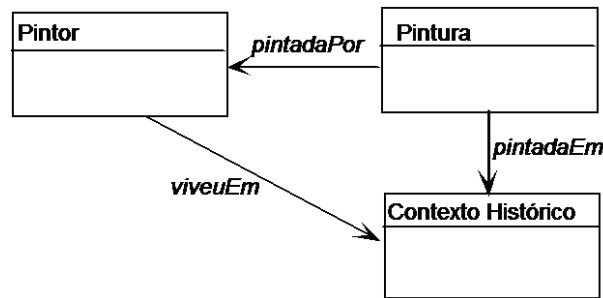


Figura 4.4: Exemplo de um esquema navegacional simples.

4.2 Modeling Language for Interaction as Conversation

A MoLIC [Sangiorgi e Barbosa 2009], *Modeling Language for Interaction as Conversation*, é uma linguagem de modelagem que representa a interface com o usuário baseada na técnica de *Storyboarding*. O fluxo de navegação do sistema é simulado a partir dos esquemas de telas.

O diagrama de interação da linguagem MoLIC representa a interação como se fosse uma conversa do usuário com o sistema. Dessa forma a MoLIC apresenta um modelo semi-estruturado, que utiliza a linguagem natural e uma estrutura organizada com notações bem definidas.

A Figura 4.5 apresenta um diagrama de diálogo básico em MoLIC. Na Figura 4.5 [Sangiorgi e Barbosa 2009] estão representados alguns elementos diagramáticos MoLIC:

- letra **a** mostra um elemento de Acesso ubíquo, que representa uma oportunidade para o usuário modificar o assunto da conversa.
- letra **b** mostra o elemento Expressão do usuário, que identifica a ação do usuário em continuar em um tópico de conversa.
- letra **c** mostra uma Cena, que é um momento de uma interação no qual o usuário decide como a conversa deve seguir. O título indica o tópico da conversa, o texto contém o diálogo.
- letra **d** mostra um Processo do sistema, que representa um processamento interno do sistema.
- letra **e** mostra uma Expressão do designer, que representa uma resposta do sistema a uma requisição do usuário.
- letra **f** mostra o Ponto de fechamento, que representa o fim da conversa.

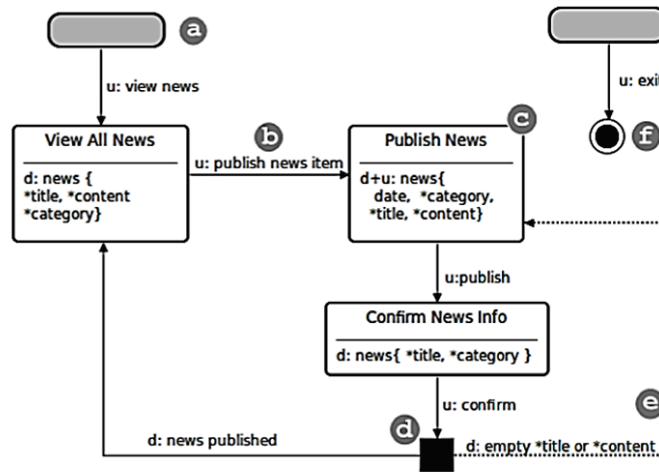


Figura 4.5: Diagrama de diálogo básico em MoLIC.

4.3 OOHDM, MoLIC e a MNLAC

O OOHDM é constituído de diversas representações gráficas para representar o esquema navegacional de um sistema interativo. Todos os conceitos são baseados no modelo conceitual, onde são definidos os objetos e classes do esquema. Essa grande variedade de representações dispense esforço de compreensão para interligar os conceitos e por fim compreender como está estruturado o modelo navegacional.

A MoLIC representa a interação como uma conversa com o usuário, apresentando diversas notações para representar essa comunicação. Da mesma forma que no OOHDM, essa diversidade de representações demanda esforço de compreensão. Os modelos MoLIC permitem apenas a representação da conversação entre usuário e sistema, e a validação é feita apenas por simulações.

A MNLAC propõe esquematizar a modelagem de fluxo de telas de um sistema interativo, focando-se apenas nesse propósito. A simplicidade do uso da *Árvore de Características* e do *Grafo de Telas* para gerar o esboço de telas do sistema é uma grande vantagem do modelo.

A representação do *Modelo Navegacional* é simples, composta apenas de duas estruturas, os vértices do modelo e as relações entre eles. Assim como o OOHDM, o modelo possui estruturas de acesso, neste trabalho chamadas de itens clicáveis. Esses itens definem o gatilho da navegação.

A MNLAC possui, ainda, a vantagem de permitir a verificação de propriedades do modelo, de forma a encontrar inconsistências, telas inatingíveis.

O OOHDM com suas diversas notações permite representar diversos tipos de informações de forma clara, por exemplo, o OOHDM possui uma notação para representar se um determinado conteúdo provém de um banco de dados. A Figura 4.6 [Barroso e Schwabe 1998] apresenta um modelo, em que um item de notícias correntes deverá ser consultado

em um banco de dados, esse item é representado por um retângulo preto na parte direita do item.

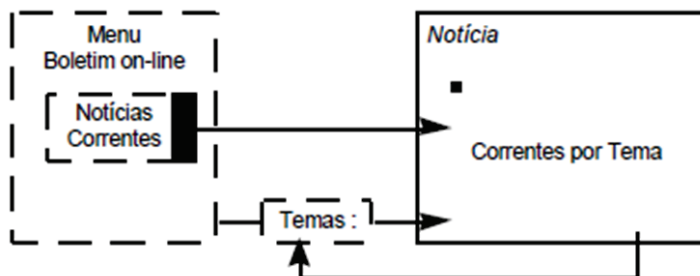


Figura 4.6: Exemplo de modelagem OOADM.

Já a MNLAC possui uma representação simples das telas e seus itens, focando apenas na interface e o fluxo navegacional, não aprofundando em detalhes de implementação das funções do sistema.

Capítulo 5

Conclusões e Trabalhos Futuros

Para projetar um bom sistema é necessário projetar uma boa interface com usuário. De nada serve um software bem planejado se os seus usuários não querem usá-lo, ou não conseguem usá-lo. Compreender as necessidades do usuário e as tarefas que precisam realizar, prever todos os aspectos e requisitos diferentes de um sistema interativo, são algumas das dificuldades ao projetar uma interface com usuário.

5.1 Contribuições

Este trabalho apresentou um método de modelagem de interfaces com usuário que visa simplificar o projeto de interfaces, focando na modelagem navegacional de sistemas computacionais com interface com usuário. É apresentado um método de modelagem que engloba desde o levantamento de requisitos, até o teste do fluxo navegacional utilizando Lógica Modal.

A *Árvore de Características* [Oliveira 2011, Oliveira et al. 2010d] apresenta uma forma simples de levantamento de requisitos, que visa facilitar a comunicação entre projetista e usuário. Utilizando esta técnica são coletadas todas as informações necessárias para o projeto da interface.

A partir da *Árvore de Características* são extraídas as informações necessárias para gerar as telas do sistema a ser projetado, representadas no *Grafo de Telas* [Oliveira 2011], que representa a ligação entre as telas do sistema.

Unindo esses dois conceitos, *Árvore de Características* e *Grafo de Telas*, este trabalho sugere a geração de esboços das telas da interface. Os esboços formam em conjunto o rascunho de como a interface será ao final do desenvolvimento, o *Storyboard* da interface.

A proposta deste trabalho é a conversão dos esboços em uma linguagem de modelagem navegacional, denominada MNLAC. O MNLAC utiliza o conceito de mundos possíveis da

Lógica Modal, neste trabalho denominados **Telas**. Uma **Tela** representa um estado da interface do sistema em um determinado momento da execução.

Definidos o conjunto de **Telas** e um conjunto de relações, utilizam-se os fundamentos da dedução em Lógica Modal para verificar a satisfação de propriedades no sistema e analisar o fluxo navegacional das telas.

Com a especificação do método MNLAC torna possível definir formalmente a modelagem navegacional de um sistema com interface com usuário. Especificamente um método que permite a verificação de propriedades no fluxo de telas, possibilitando a identificação de erros como telas inalcançáveis, problemas de usabilidade, inconsistências.

5.2 Trabalhos Futuros

No decorrer da pesquisa, alguns problemas e propostas de melhorias surgiram. Muita pesquisa ainda deve ser feita para aperfeiçoar o processo de modelagem utilizando a MNLAC. A seguir são descritas algumas otimizações e extensões que podem ser realizadas em trabalhos futuros:

- Desenvolvimento de uma ferramenta para auxiliar em todo o processo de modelagem de interface proposto neste trabalho, englobando a criação da **Árvore de Características**, geração do **Grafo de Telas**, projeto dos **Esboços de Tela**, conversão para o modelo MNLAC, e por fim a verificação de propriedades do sistema.

O processo de modelagem de interface com usuário de sistemas hipermídia apresentado neste trabalho possui a precisão que é característica de todo método formal, desta forma, é factível implementar uma ferramenta para automatizar esse processo, tornando-o mais simples e transparente aos projetistas de interface.

- Expansão do método MNLAC, de forma a oferecer maior possibilidade de representação, permitindo prever na modelagem detalhes de implementação. Permitir representar itens externos do sistema, acesso a banco de dados, validações necessárias entre mudanças de telas.
- Incorporação de métodos de verificações de propriedades, como o NuSMV, ou o MCheck.

O NuSMV¹ (“*New Symbolic Model Verifier*”) consiste em uma ferramenta Open Source que provê uma linguagem para descrever modelos que foram representados em diagramas, e verificar diretamente a validade de fórmulas LTL nestes modelos [Huth e Ryan 2008].

¹<http://nusmv.fbk.eu/>

O MCheck² é um interpretador textual simples que processa modelos de um sistema e especificações ou propriedades que um sistema deve possuir, com principal objetivo de verificar se um modelo satisfaz um conjunto de especificações [Sember 2005].

²<http://www.cs.ubc.ca/~jpseember/mcheck.html>

Referências Bibliográficas

- [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*.
- [Baba 2005] Baba, F. S., . (2005). Modelagem de Aplicações Multimídia. Disponível em: <<http://www2.dc.uel.br/nourau/document/?view=299>>.
- [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.
- [Blackburn et al. 2002] Blackburn, P., de Rijke, M., e Venema, Y. (2002). *Modal Logic*.
- [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.
- [Dix 2004] Dix, A. e. a. (2004). *Human-computer interaction*. 3.ed edition.
- [Easterbrook et al. 1997] Easterbrook, S., Lutz, R., Covington, R., Kelly, J., Ampo, Y., e Hamilton, D. (1997). Experiences Using Lightweight Formal Methods for Requirements Modeling. *NASA IV&V Facility*.
- [Garzotto e Paolini 1993] Garzotto, F. e Paolini, P. (1993). HDM A Model-Based Approach to Hipertext Application Design. In *ACM Transaction on Information Systems*, volume 11, pp. 1–26.
- [Gouveia et al. 2000] Gouveia, P., Dionísio, F. M., e Marcos, J. (2000). Lógica Computacional. Disponível em: <<http://www.dimap.ufrn.br/~jmarcos/courses/LC/Cap4.pdf>>, Acesso em: Jun, 2011.
- [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*.
- [Henrichs 2005] Henrichs, J. C. (2005). Estudo da Metodologia Orientada a Objetos OOHDm, para a Modelagem e Desenvolvimento de Websites.
- [Hix e Hartson. 1993] Hix, D. e Hartson., H. R. (1993). *Developing User Interfaces: Ensuring Usability Through Product & Process*. New York, USA.
- [Hollnagel 1993] Hollnagel, E. (1993). *Human Reliability Analysis: Context and Control*. Academic Press, London.

- [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.
- [Kieras 1999] Kieras, D. (1999). A Guide to GOMS Model Usability Evaluation using GOMSL and GLEAN3. Disponível em: <ftp.eecs.umich.edu/people/kieras>.
- [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.
- [Liskov e Berzins 1986] Liskov, B. e Berzins, V. (1986). *An Appraisal of Program Specifications*. Addison-Wesley.
- [Mandel 1997] Mandel, T. (1997). *Elements of User Interface Design*.
- [Myers 1993] Myers, B. A. (1993). *Why are Human-Computer Interfaces Difficult to Design and Implement?*.
- [Nemetz 1995] Nemetz, F. (1995). HTM: Modelagem e Projeto de Aplicações Hipermissão. Master's thesis, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.
- [Nielsen 1993] Nielsen, J. (1993). *Usability Engineering*. Boston.
- [Oliveira 2011] Oliveira, C. C. (2011). Árvore de Características e Redes de Petri Coloridas com Expressões da Lógica Proposicional: Propostas de Modelagem de Requisitos e Fluxo de Navegação. Dissertação de Mestrado.UFU/FACOM. A ser publicado.
- [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.
- [Orth 2005] Orth, A. I. (2005). *Interfaces Homem-Máquina*. Porto Alegre.
- [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.

- [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.
- [Phanouriou 2000] Phanouriou, C. (2000). UIML: A Device-Independent User Interface Markup Language. Master's thesis, Virginia Polytechnic Institute and State University.
- [Pressman 2006] Pressman, R. (2006). *Engenharia de Software*. 6ª ed. edition.
- [Puerta e Eisenstein 2001] Puerta, A. R. A. e Eisenstein, J. (2001). XIML: A Universal Language for User Interfaces. RedWhale Software. Disponível em <<http://www.miml.org/Docs.asp>>.
- [Rossi 1996] Rossi, G. (1996). *Um método Orientado a Objetos para o Projeto de Aplicações Hipermédia*. PhD thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.
- [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>.
- [Sangiorgi e Barbosa 2010] Sangiorgi, U. B. e Barbosa, S. D. (2010). Estendendo a linguagem MoLIC para o projeto conjunto de interação e interface. In *IHC10*, Belo Horizonte.
- [Schwabe 2003] Schwabe, D. (2003). *Autoria de Aplicações Hipermédia, Departamento de Informática*,. PUC Rio, Rio de Janeiro. Disponível em: <java.icmc.usp.br/books/hypermedia/ppt/OOHDM-aula1.ppt>. Acesso em: Maio, 2011.
- [Schwabe e Rossi 1995] Schwabe, D. e Rossi, G. (1995). Building hypermedia applications as navigational views of information models. In *28th Hawaii International Conference on Information Systems*, Maui, Hawaii.
- [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.
- [Sember 2005] Sember, J. (2005). MCheck: A Model Checker for LTL and CTL Formulas. Disponível em: <<http://www.cs.ubc.ca/~jpsemer/mcheck/mcheck.pdf>> , Acesso em: Junho, 2011.
- [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.
- [Sommerville 2007] Sommerville, I. (2007). *Engenharia de Software*. São Paulo, 8ª ed. edition.
- [Vilain 2002] Vilain, P. (2002). *Modelagem da interação com o usuário em aplicações hipermédia*. PhD thesis, PUC-Rio, Rio de Janeiro.