Universidade Federal de Campina Grande
Centro de Ciências e Tecnologia
Coordenação de Pós-Graduação em Informática

MEDITE^{+}: Utilizando o Processo de Roteirização para a Obtenção do Modelo de Interação EDITOR Estendido

Carlos Eduardo Caminha Lopes Rodrigues

Campina Grande - PB
Junho de 2005

Universidade Federal de Campina Grande
Centro de Ciências e Tecnologia
Coordenação de Pós-Graduação em Informática

MEDITE^{+}: Utilizando o Processo de Roteirização para a Obtenção do Modelo de Interação EDITOR Estendido

Dissertação submetida à Coordenação de Pós-Graduação em Informática do Centro de Ciências e Tecnologia da Universidade Federal de Campina Grande, Campus I como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (MSc).

Instituição: Universidade Federal de Campina Grande
Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Carlos Eduardo Caminha Lopes Rodrigues
Orientador: Bernardo Lula Júnior, Dr.

Campina Grande - PB
Junho de 2005

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

R696m Rodrigues, Carlos Eduardo Caminha Lopes
Medite + : utilizando o processo de roteirizacao para a obtencao do modelo de interacao editor estendido / Carlos Eduardo Caminha Lopes Rodrigues.- Campina Grande, 2005.
163 f. : il.

Dissertacao (Mestrado em Informatica) - Universidade Federal de Campina Grande, Centro de Ciencias e Tecnologia.

1. Interface Homem-Maquina 2. Processo de Roteirizacao 3. Dissertacao I. Lula Junior, Bernardo, Dr. II. Universidade Federal de Campina Grande - Campina Grande (PB) III. Título

CDU 004.5(043)

"MEDITE^{+}: UTILIZANDO O PROCESSO DE ROTEIRIZAÇÃO PARA A OBTENÇÃO DO MODELO DE INTERAÇÃO EDITOR ESTENDIDO"

CARLOS EDUARDO CAMINHA LOPES RODRIGUES

DISSERTAÇÃO APROVADA EM 03.06.05

PROF. BERNARDO LULA JÚNIOR, Dr.
Orientador

PROF. JOSÉ EUSTÁQUIO RANGEL DE QUEIROZ, DSc.
Examinador

PROFª MARIA ELIZABETH SUCUPIRA FURTADO, Drª
Examinadora

CAMPINA GRANDE - PB

Esta dissertação é dedicada a Deus, que me deu o dom da vida, a meus pais, a quem devo tudo que sou e a minha namorada Renata, por dividir comigo os mais belos sonhos.

Agradecimentos

Agradeço, primeiramente, a Deus, pela força, fé e esperança que tive durante a execução das atividades para que tal conclusão fosse possível.

Agradeço, carinhosamente, aos meus pais Cláudio José Lopes Rodrigues e Inês Caminha Lopes Rodrigues, pela atenção, preocupação e conselhos sempre tão seguros e úteis, bem como a todos da minha família, pelo apoio recebido.

Agradeço, também de maneira carinhosa, a minha namorada Renata Regina Barbosa Costa, pelo carinho, apoio e compreensão pelos momentos de convívio furtados para a realização deste trabalho.

Agradeço, também de maneira carinhosa, a minha irmã Cláudia Caminha Lopes Rodrigues, pelo apoio e compreensão constantes.

Agradeço, especialmente, ao meu orientador Bernardo Lula Júnior por possuir o sentimento das dificuldades que todos enfrentamos quando queremos vencer na vida profissional. Hoje posso considerá-lo grande amigo.

Agradeço, também, ao professor Eustáquio Rangel, à professora Elizabeth Furtado, pela colaboração nesse trabalho, à professora Patrícia Duarte Lima Machado além dos demais professores do Departamento de Sistemas e Computação – DSC e do Centro Universitário de João Pessoa – UNIPÊ, em especial à Professora Joseana Fechine, ao Professor Giuseppe Mongiovi e à Professora Cláudia Batista Mello que despertaram meu lado acadêmico e que me forneceram grande conhecimento.

Agradeço, ainda, aos meus grandes amigos Pablo Ribeiro Suárez, Francisco Fabian, Rodrigo César e Valéria Maria, pelas constantes demonstrações de companheirismo, prestando ajuda sempre que necessário, bem com aos demais amigos de Campina Grande e João Pessoa.

Agradeço aos meus amigos do LIA: Luana, Cheyenne, Otávio, Bruno, Raul, Ronison, Wesley, Isaac e demais companheiros do DSC.

Agradeço a todos os funcionários do DSC, em especial a Aninha, Vera e Zeneide.

Agradeço a Dona Inês, Dona Maria, Larisse e Priscila, pelos almoços, lanches cafezinhos e feijoadas às quintas-feiras.

Agradeço, portanto, a todos que contribuíram diretamente ou indiretamente para que eu concluísse mais esta etapa da vida acadêmica.

Por fim, agradeço à CAPES, pelo incentivo e recursos empregados na realização deste projeto.

Sumário

Capítulo 2 – A Metodologia MEDITE

Capítulo 3 – Extensão do Modelo EDITOR

Capítulo 4 – Representação do Componente *Diálogo* do Modelo EDITOR Estendido

Lista de Figuras

Lista de Quadros

RESUMO

Este trabalho de dissertação tem como objetivo geral definir uma nova metodologia de concepção e desenvolvimento de interfaces homem-computador – MEDITE^{+} – como variante da metodologia MEDITE pela introdução do processo de roteirização para a obtenção do modelo da interação. A introdução desta nova abordagem possibilita (i) a automatização do processo de obtenção de uma especificação conceitual parcial da interação a partir da modelagem da tarefa; e (ii) a manutenção automática da coerência entre os modelos da tarefa e da interação.

A introdução da nova abordagem em MEDITE implicou a divisão do processo de obtenção da especificação conceitual parcial da interação em dois sub-processos, a saber: geração do modelo de roteiro a partir do modelo de tarefa; e geração da especificação conceitual parcial da interação a partir do modelo de roteiro. Para a introdução dessa divisão, foi necessária a extensão do modelo de interação EDITOR (modelo EDITOR Estendido) e a definição de um formalismo (statechart) para sua representação.

A introdução do processo de roteirização aliada à definição formal (explícita) do componente de diálogo do modelo da interação permitiu a definição de um algoritmo para a obtenção automática deste componente (componente de diálogo), a partir dos elementos do modelo da tarefa e do componente de apresentação do modelo da interação.

ABSTRACT

The general aim of this dissertation is to define a new methodology on conception and development of human-computer interfaces – MEDITE+ – as a variant of the MEDITE methodology, through the introduction of the scripting process for obtaining an interaction model. The introduction of this new approach turns possible (i) the automation on the obtaining process of a partial conceptual interaction specification from the task modelling, and (ii) the automatic maintenance of the coherence between task and interaction models.

The introduction of this new approach in MEDITE implied a division of the obtaining process of a partial conceptual interaction specification into two sub-processes: scripting model generation from the task model; and partial conceptual interaction specification generation from the scripting model. For the introduction of this division it was necessary the extension on the EDITOR interaction model (modelo EDITOR Estendido) and the definition of a formalism for its representation.

The improvements mentioned above allowed the automatic generation of the partial conceptual interaction specification from the task model elements and the interaction model presentation component elements.

Capítulo 1 - Introdução

1.1 - Introdução

1.1.1 - Cenário Técnico-Científico

O advento da Informática é bastante recente na história da humanidade. O primeiro computador eletro-eletrônico data de meados do século 20 (Fedeli *et al.*, 2003). Os computadores eram muito caros, restritos a grandes centros de pesquisas, difíceis de usar, e com baixa capacidade de processamento. O Usuário do sistema interagia com os computadores por meio de painéis, ligando e desligando chaves. Não havia dispositivos de armazenamento, as rotinas (softwares) eram refeitas toda vez que necessitavam ser utilizados (não havia processos de desenvolvimentos de sistemas). Só a partir do advento dos dispositivos de armazenamento que os primeiros softwares, de maneira empírica, começaram a ser desenvolvidos. Os computadores foram barateando e tornaram-se cada vez mais acessíveis à população. Hoje, os computadores (Desktops, Notebooks, Handhelds, Celulares, entre outros aparelhos) estão presentes em praticamente todos os ambientes e sua utilização aumenta a cada dia, uma vez que o homem necessita cada vez mais desses aparelhos.

Apesar dessa crescente utilização dos sistemas computacionais, a Informática e, mais particularmente a Engenharia de Software, por ser tão recente, ainda não tem tantas soluções para os problemas envolvidos com a concepção e desenvolvimento de software como as engenharias mais "maduras" têm para os problemas que lhe concernem, como por exemplo, a Engenharia Civil. A Engenharia Civil remonta aos tempos antes de Cristo e muitos de seus problemas já vêm sendo estudados há séculos. Porém, mesmo com um histórico tão recente, avanços significativos têm sido promovidos na área da Engenharia de Software.

Muito desses avanços devem-se ao desejo do homem de automatizar sistemas, rotinas e processos de trabalho (auxiliando na realização de suas tarefas), fazendo com que os estudos em Informática se tornem cada vez mais intensos. A Engenharia de Software têm "alavancado" o processo de desenvolvimento de sistemas computacionais, estudando e propondo técnicas de programação, processos de desenvolvimento de

sistemas, linguagens, padrões, métricas, métodos formais, entre outros assuntos. Todo esse esforço é para que os projetistas de sistemas computacionais projetem sistemas mais flexíveis, modulares, reutilizáveis e compreensíveis (Gama *et al.*, 2000).

Pelo princípio da independência do diálogo (Dodani *et al.*, 1985) um sistema computacional interativo é formado por dois componentes principais, o componente funcional (aplicação) e o componente de interação (interface). Entretanto, o que se nota, é que a maior parte dos esforços no desenvolvimento de sistemas está voltada para o estudo de técnicas que melhorem cada vez mais as "funcionalidades" do sistema, relegando a interface com o usuário a um segundo plano. Quando se fala em tecnologia, sistemas e mais sistemas são analisados e suas funcionalidades (e vantagens) são propagadas aos quatro ventos. Costuma-se sempre exaltar essa ou aquela funcionalidade e o que foi implementado a mais em relação à versão anterior. Entretanto, na construção de tais sistemas, na maioria das vezes, os projetistas não levam em conta o elemento principal: o usuário. De nada adianta um sistema com funcionalidades que possam resolver todos os problemas do mundo, se não há uma interface que permita que os usuários possam acessá-las e utilizá-las com eficácia. Se a interface entre ambos, usuário e funcionalidade, não for bem projetada, o sistema, que deveria auxiliar o usuário na execução de suas tarefas de forma espontânea e natural, não irá ajudá-lo. Pelo contrário, o sistema poderá até mesmo ser descartado pelo usuário. A verdade é que na concepção de sistemas de informação os usuários que deveriam ser o centro das atenções, terminam virando meros coadjuvantes.

Se observados, os processos de desenvolvimento de software, quase em sua totalidade, não levam em consideração ou não dão a devida importância à concepção da interface com o usuário (Vasconcelos, 2004). Outro indício dessa situação é que o projeto e a concepção de interfaces não fazem parte da formação dos desenvolvedores de software. Apesar dessa "renegação", diversos estudos na área têm sido promovidos.

A concepção de Interfaces Homem-Computador – assim como o desenvolvimento do software em si –, começou sua trajetória de forma empírica. Não existiam estudos que auxiliassem o projetista no desenvolvimento da interface do sistema, sendo elas construídas sem uma preocupação maior quer com o perfil do usuário quer com a tarefa a ser realizada com o auxílio do sistema. Nesse tipo de abordagem, o uso do conhecimento ergonômico ou qualquer outro tipo de conhecimento envolvido no processo de desenvolvimento de interfaces é apenas usado na fase de avaliação dos

protótipos acarretando, assim, um freqüente re-projeto (Suárez *et al.*, 2004). Somente a partir de trabalhos desenvolvidos nos campos da Psicologia Cognitiva e Ergonomia (LOGO (Papert, 1980), Norman (Teoria da Ação) (Norman, 1983), Card&Moran (Modelo do processador humano) (Card, *et. Al.*, 1983) , Scapin (Guia ergonômico) (Scapin, 1987) , etc), questões relativas ao usuário e à tarefa começaram a ser levadas em consideração nos projetos de interfaces dos sistemas.

A abordagem baseada na análise da tarefa do usuário surge como uma alternativa ao processo empírico de concepção de interfaces. O conhecimento ergonômico e as experiências de projeto são utilizados desde os primeiros estágios da concepção. Várias metodologias de concepção de interfaces baseadas na análise da tarefa foram propostas, entre as quais podem ser citadas:

- **ADEPT** (Advanced Design Environment for Prototyping with Tasks) (Johnson et al., 1993; Markopoulos et al., 1992; Wilson et al., 1993);
- **ALACIE** (Atelier Logiciel d'Aide à la Conception d'Interfaces Ergonomiques) (Gamboa e Scapin, 1997; Gamboa, 1998);
- **ERGOSTART** (methodolgiE oRientée erGonomie du lOgiciel: depuiS la description des Tâches utilisAteurs jusqu'à la Realisation d'InTerface) (Hammouche, 1995);
- **MACIA** (Metodologia de Assistência à Concepção e à realização de Interfaces Adaptadas) (Furtado, 1997; Furtado, 1999);
- **MCIE** (Método para Concepção de Interfaces Ergonômicas) (Turnell, 2004);
- **MEDITE** (MAD* + EDITOR + ERGONOMIA) (Guerrero e Lula, 2001; Guerrero e Lula, 2002; Guerrero, 2002a); e
- **TRIDENT** (Tools foR an Interactive Development EnvironmeNT) (Bodart e Vanderdonckt, 1993; Bodart et al., 1994; Bodart et al., 1995).

Essas metodologias são baseadas na análise da tarefa do usuário e dirigidas por modelos (de apresentação, de diálogo, da interação, do usuário, de arquitetura, de aplicação, de domínio - não necessariamente presentes em todas as metodologias) para auxiliar o(s) projetista(s) nas atividades de concepção de interfaces. O conhecimento (ergonômico e/ou de projeto) referente à obtenção de uma especificação conceitual da interação é representado na forma de regras que relacionam elementos presentes nas

descrições da tarefa e do usuário com elementos presentes na descrição da interação (Suárez, 2004). Em suma, o que todas elas pretendem é auxiliar o projetista na aquisição da descrição da interação (modelo da interação) a partir da descrição da tarefa do usuário (modelo da tarefa), usando o conhecimento representado na forma de regras.

Entretanto, alguns problemas ainda perduram no que diz respeito à utilização dessas metodologias e ao próprio tipo de abordagem utilizada (formalização e utilização do conhecimento em forma de regras) para obtenção da interação a partir da descrição da tarefa do usuário (Guerrero, 2002a; Suárez, 2004). Problemas como: necessidade de algum conhecimento ou experiência no campo da ergonomia, ausência de ferramentas computacionais, dificuldade na escolha e na utilização de regras ergonômicas, dificuldade de se passar do modelo da tarefa para um modelo de interação particular, manutenção da coerência entre os modelos, construção e manutenção da bases de conhecimento, etc. Além de todos os problemas supracitados, a não uniformidade de termos e conceitos no âmbito da concepção de IHC é mais um fator que dificulta a utilização efetiva de quaisquer metodologias pelos projetistas. Esses problemas foram constatados diretamente nos experimentos realizados no âmbito da UFCG pelo Grupo de Interfaces Homem-Máquina (GIHM) em concepção e desenvolvimento de interfaces utilizando a metodologia MEDITE (Rodrigues e Lula, 2004).

Para analisar esses problemas, Suaréz (Suárez, 2004) utilizou métodos de gestão do conhecimento no âmbito das diversas metodologias de concepção de IHC citadas com objetivo de construir uma ontologia para unificar a grande diversidade de conceitos presentes e propôs uma nova abordagem para o processo de obtenção de uma especificação conceitual da interação a partir das descrições da tarefa e do usuário que eliminam alguns desses problemas.

1.1.2 - Uma Nova Abordagem na Concepção de Interfaces

Como mencionado anteriormente, as metodologias presentes no cenário atual utilizam em sua maioria o conhecimento ergonômico na forma de regras para a transformação do modelo da tarefa no modelo da interação. Estudos apontam que um dos maiores problemas nas metodologias de concepção de IHC está justamente na transformação do modelo de tarefa para o modelo de interação (van Welie, 2001).

Adicionando-os às dificuldades da utilização do conhecimento codificado em forma de regras (Vanderdonckt, 1995)**,** (Vanderdonckt e Bodart, 1994), (Hammouche, 1995) e (Barbosa *et al.*, 2002), tornou-se necessário um esforço de pesquisa para a minimização desses problemas. Suárez (Suárez, 2004) mostrou que é possível obter de forma automática o modelo conceitual da interação a partir do modelo da tarefa, preservando a decomposição estrutural e temporal da tarefa. Um dos resultados do seu trabalho foi à concepção de três meta-modelos – tarefa, usuário e interação – que compõem a ontologia proposta. A classificação e a representação dos conhecimentos obtidos nos meta-modelos de tarefa, usuário e interação, contudo, não foram fatores suficientes para garantir uma solução para a dificuldade de integração e de utilização desses meta-modelos no processo de obtenção de uma especificação conceitual da interação. Para tanto, o autor propôs, então, o *meta-modelo de roteiro* (*script*), baseado em uma abordagem fundamentada em uma metáfora cênica, visando facilitar (e mesmo automatizar) o processo de obtenção do modelo da interação a partir do modelo de tarefa e possibilitar a manutenção automática da coerência entre os meta-modelos (rastreabilidade).

O meta-modelo de roteiro é, portanto, a base da nova abordagem para obtenção da especificação parcial da interação a partir da modelagem da tarefa.

1.2. – Objetivos do Trabalho

1.2.1 - Objetivo Geral

O estudo levado a efeito por Suárez (Suárez, 2004) foi independente de qualquer metodologia específica de concepção de interfaces homem-computador. O objetivo geral o presente trabalho de dissertação é definir uma variante da metodologia de concepção e desenvolvimento de interfaces homem-computador MEDITE, pela introdução da abordagem proposta por Suárez (Suárez, 2004) para a obtenção do modelo da interação. Este variante chamar-se-á MEDITE^{+}. A introdução dessa nova abordagem possibilitará (i) a automatização do processo de obtenção de uma especificação conceitual parcial da interação (modelo EDITOR (Lula, 1992)) a partir da modelagem da tarefa (Modelo TAOS (Medeiros, 1995)), e (ii) a manutenção automática

da coerência entre os modelos da tarefa e da interação. Esse propósito geral será alcançado com a realização dos objetivos específicos descritos a seguir.

1.2.2 - Objetivos Específicos

1.2.2.1 - Extensão do Modelo de Interação EDITOR e sua Representação

MEDITE define como modelo da interação o modelo EDITOR desenvolvido por Lula (Lula, 1992). Embora o modelo EDITOR seja adequado à representação do componente apresentação da interface do sistema, ele carece de uma definição explicita do componente diálogo da aplicação, atualmente, a definição do diálogo em MEDITE passa pela definição de métodos de envio de mensagens entre objetos (OO), ou seja, exige do projetista atividades em nível de programação. Pretende-se, pois, estender e definir uma representação conveniente para o modelo EDITOR (EDITOR Estendido), de forma a representar adequadamente (explicitamente) tanto a apresentação quanto o diálogo da aplicação, permitindo, então, que o projetista atue no nível de abstração apropriado.

1.2.2.2 - Introdução do Modelo de Roteiro

A introdução da nova abordagem em MEDITE implicará a divisão do processo de obtenção da especificação conceitual parcial da interação em dois sub-processos, a saber: especificação do modelo de roteiro (Modelo de Roteiro); e especificação conceitual parcial da interação (Modelo EDITOR Estendido). Ou seja, a substituição do processo de derivação do modelo da interação utilizando regras ergonômicas pelo "processo de roteirização" (modelo de roteiro), implicando a definição de uma variante da metodologia baseada na nova abordagem (MEDITE^{+}).

1.2.2.3 - Definição e Implementação do Processo de Roteirização

Definição e implementação dos algoritmos de transformação do modelo da tarefa (TAOS) no modelo de roteiro e do modelo de roteiro no modelo de interação (EDITOR Estendido).

1.2.2.4 – Definição e Implementação do Mecanismo de Manutenção da Coerência entre os Modelos de Interação e da Tarefa

A solução preconizada por Suárez (2004) define a rastreabilidade entre elementos de modelos distintos o que permite o mapeamento de elementos de um modelo em outro. Pretende-se, pois, definir e implementar os mecanismos necessários para que se possa manter a coerência entre os modelos.

1.3 - Relevância

O projeto de interface é parte integrante e importante do processo de desenvolvimento de software. Cada grupo (ou corporação) necessita utilizar uma metodologia para o desenvolvimento de interfaces. MEDITE é uma metodologia que vem sendo refinada e utilizada no contexto local, no intuito de ajudar o projetista na realização deste componente tão importante do software. As definições e mudanças propostas neste trabalho permitirão que o(s) projetista(s) de interfaces que atualmente utiliza(m) MEDITE obtenha(m) de forma automática uma especificação (parcial) do modelo da interação (Modelo EDITOR Estendido) a partir de uma especificação da tarefa segundo o modelo TAOS. Ou seja, de posse do modelo da tarefa do usuário, o projetista terá um primeiro esboço da estrutura da interface de maneira automática. De imediato, haverá uma diminuição na carga despendida pelo projetista para a obtenção desse artefato. A extensão do modelo EDITOR possibilitará ao projetista a verificação de informações referentes ao diálogo da aplicação sem a necessidade da implementação,

ou seja, em nível conceitual. Por fim, havendo a necessidade ou desejo de alteração (por parte do projetista e/ou usuário) no modelo parcial da interação, haverá a manutenção automática da coerência entre os modelos (mecanismos de manutenção de coerência), uma vez que a inserção do modelo de roteiro permite o rastreamento entre elementos de modelos distintos, fazendo com que o projeto contenha modelos sempre compatíveis. Haverá, portanto, maior ganho de produtividade, pois o projetista ganhará mais agilidade, fontes de informações e ferramentas de trabalho. Estes pontos já são de fundamental relevância à realização do trabalho.

1.4 - Metodologia de Desenvolvimento

Esse trabalho utilizou uma metodologia de desenvolvimento baseada nas seguintes etapas:

1. Revisão bibliográfica das metodologias de concepção de interfaces e da nova abordagem preconizada por Suárez (2004).
2. Análise da metodologia MEDITE, visando a identificação e catalogação dos problemas associados tanto à sua utilização quanto à abordagem de obtenção do modelo de interação por ela utilizada.
3. Estudo de formalismos baseados em eventos/transição e definição de um formalismo apropriado para a representação do modelo EDITOR Estendido.
4. Definição da metodologia MEDITE^{+} baseada no processo de roteirização.
5. Estudo de mecanismos para a manutenção da coerência entre os modelos.
6. Projeto e implementação do algoritmo de transformação de um modelo de tarefa TAOS em um modelo de interação EDITOR Estendido.
7. Projeto e implementação de algoritmos de manutenção da coerência.
8. Elaboração de estudos de caso para a validação das alterações propostas.

1.5 - Estrutura da Dissertação

A Dissertação está dividida em 8 capítulos. O capítulo 1 contém as informações necessárias ao entendimento do que representa a pesquisa, quais sejam, cenário-técnico científico, objetivos do trabalho, suas justificativas (relevância) e a sua metodologia de realização.

O capítulo 2 apresenta uma descrição da metodologia MEDITE, uma breve comparação de MEDITE com as demais metodologias de desenvolvimento de IHC e os principais problemas encontrados em sua utilização.

O capítulo 3 aborda a importância e vantagens da utilização de modelos para a representação do diálogo da interação de um sistema. Apresenta o modelo de interação utilizado na metodologia MEDITE – EDITOR –, suas características e seus elementos. Aborda as dificuldades encontradas na sua atual composição e, por fim, apresenta uma proposta de extensão do modelo, para que o mesmo contenha um componente explícito de representação do diálogo da interação.

O capítulo 4 é referente à escolha de um formalismo apropriado para a representação do modelo EDITOR Estendido. Características e requisitos para a escolha do formalismo de representação do diálogo da interação são apresentados. Por fim, é apresentado o formalismo escolhido para a representação do componente *Diálogo* do modelo EDITOR Estendido.

O capítulo 5 apresenta o processo de roteirização desenvolvido por Suárez (Suárez, 2004) e a sua introdução em MEDITE, definindo MEDITE^{+}. Este capítulo apresenta também os requisitos, as funcionalidades básicas e os algoritmos necessários à manutenção da coerência entre os modelos.

O capítulo 6 apresenta o projeto e implementação dos modelos de roteiro e interação, dos algoritmos de obtenção do modelo da interação, dos mecanismos de manutenção da coerência e do componente de diálogo da aplicação.

O capítulo 7 apresenta a análise dos resultados obtidos com a utilização de MEDITE^{+} em um estudo de caso para validação das modificações propostas.

Por fim, o capítulo 8 apresenta as conclusões da pesquisa e sugestões para trabalhos futuros.

Capítulo 2 - A Metodologia MEDITE

Este capítulo descreve a metodologia MEDITE, a motivação de sua concepção e sua descrição original. Em seguida, são apresentadas as dificuldades e limitações encontradas com a utilização da metodologia MEDITE, assim como as dificuldades e limitações inerentes ao tipo de abordagem preconizada por algumas metodologias de concepção de interfaces que se baseiam no modelo de tarefa - derivação do modelo da interação a partir do modelo de tarefa através de regras ergonômicas e/ou de produção.

2.1 – Introdução

Como mencionado no capítulo 1, o projeto de interfaces do usuário passou de um estágio de desenvolvimento empírico para uma abordagem mais madura, que leva em consideração os aspectos referentes às características dos usuários e das tarefas a serem realizadas com o auxílio do sistema. Os aspectos de Psicologia Cognitivas e Ergonomia (LOGO (Papert, 1980), Norman (Teoria da Ação) (Norman, 1983), Card&Moran (Modelo do processador humano) (Card, *et. Al.*, 1983), Scapin (Guia ergonômico) (Scapin, 1987), etc) se tornaram fundamentais e passaram a ser utilizados desde o início do processo de concepção para a obtenção de interfaces de qualidade. Diversas metodologias foram desenvolvidas levando em conta esses aspectos e incorporando modelos representativos desses aspectos. Surgiram, então, as metodologias (ou ambientes) de desenvolvimento de interfaces baseadas em modelos (Silva, 2000). Para Silva (Silva, 2000), as principais vantagens de se utilizar modelos para interfaces com usuários são:

- Os modelos podem prover uma descrição mais abstrata da interface com o usuário do que por outros tipos de metodologias (ferramentas) de desenvolvimento não baseadas em modelos.
- Eles facilitam a criação de métodos de projeto e implementação da interface de uma maneira sistemática, uma vez que oferecem a possibilidade: (1) de

modelagem da interface com o usuário em diferentes níveis de abstração; (2) de refinamentos incrementais das representações; e (3) de reuso de especificações de interfaces.

- Eles provêem a infraestrutura necessária para automatizar tarefas relacionadas com o processo de projeto e implementação das interfaces.

Guerrero (Guerrero, 2002) cita outras vantagens com o uso de modelos:

- A possibilidade de uma fácil automatização do processo através de ferramentas específicas
- Consistência e reutilização
- A possibilidade de desenvolvimento interativo.

Quando o processo de concepção tem como elemento base à integração da representação da tarefa do usuário, ou seja, o modelo de tarefa, fala-se da abordagem baseada na tarefa. Esta é uma especialização da abordagem baseada em modelos, com a particularidade de que o principal modelo utilizado é o da tarefa do usuário (Gamboa, 1998).

As metodologias que se baseiam no modelo da tarefa do usuário constituem um expressivo apoio à concepção de interfaces ergonômicas. Elas permitem, a partir da descrição da tarefa, do perfil do usuário e de princípios ergonômicos, a construção de uma especificação executável (protótipo) da interface levando em conta os objetivos do usuário (Guerrero, 2002). Como exemplos significativos desse tipo de metodologia podemos citar: ADEPT, ALACIE, ERGO-START, MACIA, MEDITE, MCI, TRIDENT, entre outras.

Porém, o que se nota é que, mesmo com o auxílio das metodologias de desenvolvimento baseadas na tarefa do usuário e em modelos, o projetista de interfaces continua a se defrontar com inúmeras dificuldades relativas ao uso dos conhecimentos relacionados aos aspectos do usuário e da tarefa. Guerrero, em (Guerrero e Lula, 2001; Guerrero e Lula, 2002; Guerrero, 2002), relaciona as seguintes dificuldades na utilização dessas metodologias:

- "(...) Necessidade de experiência ou conhecimento em Ergonomia;
- Dificuldades de escolha e de utilização de regras ergonômicas em relação às etapas do processo. Em geral, a taxonomia das regras não tem uma relação única com as etapas do processo;

- Dificuldades de passagem / transformação do modelo da tarefa para um modelo conceitual de interação;
- Dificuldades na utilização e aplicação da metodologia na ausência das ferramentas;
- Dificuldades em modificar o código da interface (...)."

As hipóteses levantadas por Guerrero (2002) em sua pesquisa sobre as causas dos problemas supracitados, seriam: ausência de um relacionamento claro entre o modelo da tarefa e o modelo da interação definidos nessas metodologias; ausência de uma descrição gráfica adequada do modelo de interação utilizado que induzisse uma visualização mental da estrutura da apresentação da interface; e, por fim, ausência de um modelo explícito capaz de representar a arquitetura do código da interface. Para atingir seu objetivo principal (construção de uma metodologia capaz de minimizar os problemas anteriormente apresentados) Guerrero (2002) propôs a metodologia MEDITE.

2.2 - A Metodologia MEDITE

MEDITE (MAD* + EDITOR + ERGONOMIA) surgiu como uma proposta de auxiliar o projetista (principalmente aquele que não tem conhecimento sobre Ergonomia ou a equipe de desenvolvimento que não dispõe de ergonomistas) no processo de especificação de interfaces ergonômicas ou que agreguem um grau elevado de conhecimento ergonômico (Guerrero, 2002). MEDITE é uma metodologia de concepção de IHC baseada na tarefa e orientada a modelos.

A modelagem da tarefa do usuário é feita através do formalismo MAD* (Hammouche, 1995; Gamboa, 1998) e utiliza o modelo EDITOR (Lula, 1992) como modelo unificado de interação e de arquitetura. Segundo Guerrero (2002), **MEDITE** define o processo de construção de interfaces em 5 etapas:

(i) "(...) Análise e modelagem da tarefa - Serão identificadas as tarefas, e em seguida, será feita a modelagem das mesmas com a construção da árvore de tarefa e o preenchimento dos respectivos descritores. Esta etapa tem como entrada os dados sobre o usuário e o domínio da tarefa. O produto gerado no

final desta etapa é a descrição MAD* da tarefa (árvore MAD* e seus descritores);

(ii) Especificação conceitual parcial da interação (construção das Árvores EDITOR) - Esta etapa consiste em produzir a especificação conceitual inicial da interação. Trata-se do processo de construção das árvores EDITOR, ou seja, é nesta etapa que cada agente do Modelo EDITOR (Editor, Visão e Objeto_de_Interação) é definido. Esta etapa tem como entrada a descrição MAD* gerada na etapa anterior e é o momento da primeira inserção do conhecimento ergonômico, aqui representado sob a forma de regras de produção. O processo de transformação da árvore MAD* em árvore EDITOR é realizado e conceitualizado por meio das Regras Ergonômicas para Construção da Árvore EDITOR. São especificados neste momento aspectos relacionados com a estrutura, o sequenciamento, estilos de interação e layout das telas ou janelas. Nesta fase o projetista tem uma visão geral e inicial (esboço) das telas ou janelas da interface a ser construída;

(iii) Especificação completa (definição dos atributos) - A 3ª etapa consiste na definição dos atributos das árvores EDITOR. Esta etapa tem como entrada a árvore gerada na etapa anterior e a descrição MAD* da tarefa gerada na primeira etapa. Nesta fase, ocorre a segunda inserção do conhecimento ergonômico (as Regras Ergonômicas para Definição dos Atributos) a árvore EDITOR é complementada, ou seja, são definidos os atributos de cada uma das facetas: Apresentação (localização, formato, tipo, tamanho de fonte, etc.); Abstração (com relação ao domínio da aplicação) e; Controle que define o encadeamento do diálogo (inter e intra Editores).;

(iv) Geração do protótipo - Esta etapa consiste na geração do protótipo (código da interface) a partir das árvores EDITOR vindas da etapa anterior. O modelo de arquitetura espelha exatamente o modelo conceitual de interação. e

(v) Avaliação - esta atividade é distribuída em todas as etapas anteriores da metodologia e consiste na avaliação de cada produto de cada etapa (...)".

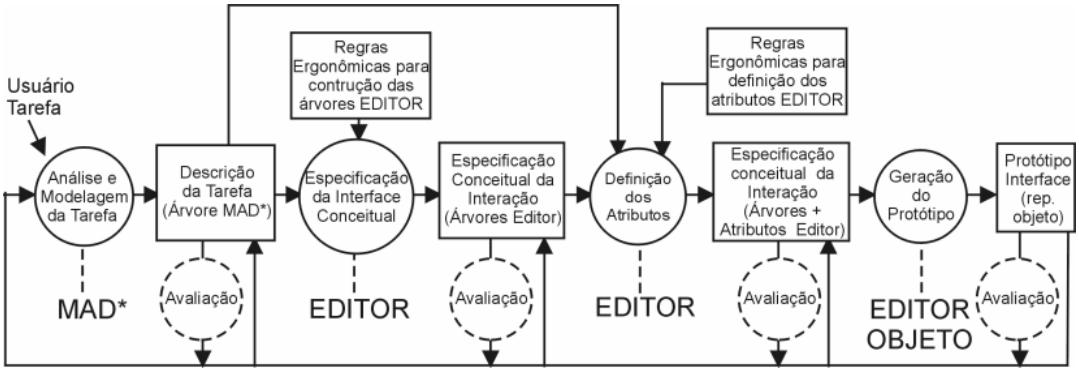


Figura (2.1): Metodologia MEDITE

MEDITE propôs, originalmente, a utilização do formalismo MAD* para a análise e modelagem da tarefa (etapa 1). A partir do trabalho de Kafure (Kafure, 2000), que validou o formalismo TAOS (Medeiros, 1995; Medeiros e Rousselot, 1995a; Medeiros e Rousselot, 1995b; Medeiros e Rousselot, 1995c; Medeiros *et al.*, 2000) com respeito ao MAD (Scapin, 1989) e dos trabalhos de Medeiros (Medeiros, 2003) e Cordeiro (Cordeiro, 2003) que culminaram no desenvolvimento da ferramenta iTAOS (Medeiros et al., 2002a ; Medeiros et al., 2002b ; Medeiros et al., 2002c ; Medeiros et al., 2002d), o formalismo TAOS também passou a ser utilizado nessa etapa da metodologia e atualmente está se tornando o padrão da metodologia para a análise e modelagem da tarefa. Mais informações sobre o formalismo TAOS o leitor deve consultar (Medeiros *et al.*, 2000) ou (Kafure, 2000).

2.3 - Dificuldades Encontradas na Utilização de MEDITE

MEDITE foi utilizada inicialmente no desenvolvimento de um tutorial na web para auxiliar o ensino da disciplina Teoria da Computação (tópico relativo à Máquina de Turing) e posteriormente no desenvolvimento da ferramenta iTAOS. No momento da concepção da ferramenta iTAOS, no entanto, algumas dificuldades formam relatadas pelos projetistas. Segundo Medeiros (Medeiros, 2003), "as regras ergonômicas disponíveis na primeira versão da base de regras ergonômicas não foram suficientes para que fosse realizada a transformação de toda a árvore da tarefa nas árvores EDITOR correspondentes. Por esse motivo, algumas regras foram modificadas e outras criadas baseando-se na nossa experiência com outras ferramentas equivalentes já existentes."

Apesar do sucesso do seu emprego na concepção daqueles sistemas, sua utilização apresentou outras dificuldades relacionadas à (i) obtenção do modelo da interação a partir do modelo da tarefa (utilização de regras ergonômicas), na (ii) definição do aspecto diálogo da interação (modelo de interação EDITOR), (iii) ausência de mecanismos de manutenção da compatibilidade dos modelos após a alteração no modelo da interação e, por fim, (iv) ausência de ferramentas que auxiliassem o projetista na aquisição do modelo da interação (Rodrigues e Lula, 2004).

2.3.1 - Utilização de Regras Ergonômicas

Apesar da existência de regras para guiar a construção do modelo de interação (EDITOR) a partir do modelo da tarefa (MAD), a utilização dessas regras não é das mais simples. Além do mais, algumas regras exigem conhecimentos do projetista que não estão expressos na modelagem da tarefa, deixando a sua aplicação sujeita à interpretação do projetista.

O processo de obtenção da especificação (parcial) da interação mediante o uso de regras nem sempre é um processo fácil para o projetista. Bodart e Vanderdonckt (Vanderdonckt e Bodart, 1994), Hammouche (Hammouche, 1995) e Barbosa (Barbosa et al., 2002) apontam algumas dificuldades no uso do conhecimento ergonômico e/ou de projeto representados na forma de regras ergonômicas para a obtenção da descrição do modelo conceitual da interação a partir das descrições da tarefa e do perfil do usuário do sistema. Além da dificuldade introduzida, muitas vezes, pela necessidade de interpretação das regras por parte do projetista, outras dificuldades apontadas dizem respeito à construção e manutenção da(s) bases(s) de conhecimento, à ausência de mecanismos de manutenção da coerência entre os modelos envolvidos, à possibilidade de regras contraditórias, entre outras. Todas as metodologias anteriormente mencionadas (ADEPT, ALACIE, ERGO-START, MACIA, MCI, MEDITE, TRIDENT) têm em comum o conhecimento para a construção de um modelo conceitual da interação baseado em forma de regras/tabelas (ergonômicas/produção). Ou seja, esses problemas não estão presentes apenas em MEDITE, mas em todas as metodologias que utilizam a mesma abordagem.

Para melhor exemplificar tal fato em MEDITE, algumas regras serão analisadas a seguir.

"Regra 04: SE as sub-tarefas forem ligadas pelo construtor SEQ (MAD*), ENTÃO definir uma *Visão* (EDITOR) apenas para a primeira sub-tarefa da seqüência. As demais sub-tarefas tomarão o lugar de uma outra *Visão* ou a *Visão* correspondente à primeira sub-tarefa. (depende do diálogo)".

Como expresso no próprio texto da regra, a utilização desta depende do diálogo da aplicação. O modelo EDITOR não conta com uma representação explícita do componente de diálogo da aplicação. Ora, se este componente não é explicitado no modelo, como o projetista ou um mecanismo de automação irá resolver tal dependência? Não fica claro qual o tipo de dependência em questão.

"Regra 16: SE as tarefas implicam ações similares (MAD*) ENTÃO propor procedimentos similares de acesso às opções de menus. (atributos EDITOR - diálogo)".

Não há no modelo da tarefa nenhuma informação sobre a similaridade de tarefas. Essa regra, portanto, requer uma interpretação por parte do projetista, sendo impossível sua automatização da maneira como está expressa.

2.3.2 - Definição do Aspecto Diálogo da Interação (modelo de interação EDITOR)

O modelo de interação EDITOR, utilizado em MEDITE, define explicitamente apenas o aspecto da apresentação da interação, deixando a cargo do projetista a definição do diálogo através de mensagens enviadas aos objetos de interação envolvidos. O projetista precisa descrever precisamente o comportamento externo do sistema, sem ter que se preocupar com detalhes de implementação (Tse e Pong, 1991). A ausência de uma representação explicita do componente de diálogo da aplicação, obriga o projetista a atuar no nível de implementação, ou seja, em um nível de abstração muito baixo. Portanto, a falta do componente (explícito) do diálogo inviabiliza a checagem da ausência de inconsistências e incoerências, prejudica a visualização do comportamento do sistema, prejudica a visualização da hierarquia e da concorrência entre elementos da interface, dificulta a comunicação entre projetista(s) e usuário(s), entre outras dificuldades. Ou seja, o projetista só saberá como o sistema irá funcionar

depois que implementá-lo. Caso haja um erro ou inconsistência, haverá necessidade de um re-projeto, acarretando um desperdício de tempo e recursos, assim como demandando um maior esforço de sua parte.

2.3.3 - Ausência de Mecanismos de Manutenção da Compatibilidade dos Modelos Após a Alteração no Modelo da Interação.

No atual formato da metodologia, não há como relacionar elementos de modelos distintos. Por exemplo, não há como saber de qual tarefa ou sub-tarefa uma determinada janela ou sub-janela foi gerada. Além do mais, havendo necessidade de mudanças no modelo de interação – um processo natural na avaliação da interface – não há como o projetista manter a compatibilidade com os modelos anteriores, uma vez que não há como identificar qual tarefa estava associada com o elemento da interação modificado. No caso da inserção de um novo elemento no modelo da interação (um *Editor*, uma *Visão*, um *Objeto de Interação*), esse elemento não contará com a sua contra-parte na modelagem da tarefa, pois não há como identificar nem como inserir o elemento correspondente no modelo da tarefa.

2.3.4 - Ausência de Ferramentas para Auxílio ao Projetista na Aquisição do Modelo da Interação

A ausência de ferramentas de auxílio a aquisição do modelo da interação representa um esforço adicional requerido do projetista. Embora o modelo EDITOR possa ser representado sem o auxílio ferramental, esse trabalho é intenso e exaustivo, principalmente para sistemas de grande porte. Caso haja a necessidade de modificações no modelo da tarefa, toda a estrutura do modelo da interação deverá ser re-analisada, uma tarefa árdua se realizada manualmente!

Essas dificuldades encontradas na utilização de MEDITE foram corroboradas pelos alunos matriculados na disciplina Inteligência Artificial, oferecida no período 2003.1 pela Coordenação de Pós-Graduação em Informática (Copin) da UFCG, que implementaram um protótipo de sistema especialista para automatizar a atividade de obtenção de uma especificação conceitual da interação a partir das descrições da tarefa e

do usuário, com o intuito de demonstrar como seria possível auxiliar o projetista na realização dessa atividade (Suárez et al., 2003b).

O protótipo do sistema foi implementado em *JESS* (*Java Expert System Shell*) (Jess, 2003), um *shell* para o desenvolvimento de sistemas especialistas desenvolvido na plataforma *Java* (JAVA, 2003). A implementação baseou-se em um seleto e restrito conjunto de regras propostas em MEDITE. Um total de vinte regras foi selecionado para compor a base de regras implementada no protótipo.

A implementação das regras ergonômicas no ambiente *JESS* permitiu a geração automática de uma descrição conceitual da interação, segundo o modelo de interação EDITOR adotado pela metodologia MEDITE. Entretanto, apesar do estudo de caso demonstrar a viabilidade e utilidade do sistema de apoio ao projetista na concepção de interfaces, foram observadas (Suárez et al., 2003b ; Suárez, 2004) grandes dificuldades no uso dessa abordagem no processo de concepção de interfaces com o usuário em relação à construção da base de conhecimento, representação do conhecimento ergonômico, abrangência das regras e manutenção da coerência entre os modelos.

Esse conjunto de dificuldades encontradas a partir da utilização da metodologia MEDITE em diversos contextos de utilização motivou a realização de pesquisas para identificação das fontes daquelas dificuldades e a proposição de elementos de solução.

2.4 - Conclusão

Diante das dificuldades constatadas, quais sejam: (1) obtenção do modelo da interação a partir do modelo da tarefa, utilizando-se regras ergonômicas; (2) ausência do componente (explícito) de representação do diálogo do modelo de interação EDITOR; (3) ausência de mecanismos de manutenção da compatibilidade dos modelos após a alteração no modelo da interação; e (4) ausência de ferramentas de auxílio ao projetista na aquisição do modelo da interação, torna-se necessário, portanto, desenvolver soluções para esses problemas.

O capítulo 3 e o capítulo 4 tratam da questão relativa à extensão do modelo de interação EDITOR para que o mesmo conte com o componente – explícito – do diálogo e de sua representação, respectivamente.

O capítulo 5 apresenta a metodologia MEDITE^{+}, uma derivação da metodologia MEDITE com a utilização da nova abordagem para a obtenção do modelo parcial da interação a partir do modelo da tarefa. Tal abordagem permite a rastreabilidade entre os

modelos, o que possibilita a construção de mecanismos de manutenção de coerência entre elementos de modelos distintos.

O capítulo 6 trata dos aspectos de projeto e implementação dos algoritmos de obtenção do modelo da interação EDITOR e de manutenção de coerência entre os modelos de tarefa e de interação.

Face à existência da ferramenta de modelagem da tarefa iTAOS, o formalismo TAOS passa a ser empregado doravante na modelagem da tarefa da metodologia MEDITE^{+}.

Capítulo 3 – Extensão do Modelo EDITOR

Este capítulo, primeiramente, aborda a importância e vantagens da utilização de modelos para a representação do diálogo da interação de um sistema. Em seguida, apresenta as características do modelo de interação utilizado na metodologia MEDITE – EDITOR –, suas características, seus elementos. Aborda as dificuldades encontradas na sua atual composição e, por fim, apresenta uma proposta de extensão do modelo, para que o mesmo contenha um componente de representação do diálogo da interação.

3.1 - Introdução

Vanderdonckt (Vanderdonckt *et al.*, 2003) define duas classes de modelos presentes nas metodologias baseadas na tarefa do usuário, a saber: **modelos abstratos** e **modelos concretos**. Entre os modelos abstratos estão o *modelo da tarefa* (descreve os objetivos que o usuário pretende alcançar, e as ações que ele tem que realizar para alcançar tais objetivos), o *modelo de domínio* (descreve os objetos e dados que o usuário irá lidar) e o *modelo do usuário* (descreve propriedades dos usuários, como o seu nível de experiência com determinado sistema). Já os modelos concretos representam elementos que possuem uma ligação direta com o mundo físico. *O modelo de apresentação* descreve a aparência visual da interface. Na apresentação é especificado como os objetos de interação (widgtes) são selecionados, onde serão distribuídos, entre outras coisas. O *modelo de diálogo* (ou *de navegação*) descreve o fluxo de troca de informação entre o usuário e o sistema. Aqui, são definidas a estrutura navegacional da interface (estrutura do diálogo) e as técnicas de interação (widgets, objetos de interação) usadas.

Segundo Elwert *et al.* (Elwert *et al*., 1996) a utilização de especificações formais na concepção de interfaces homem-computador, permite, entre outras coisas:

- a descrição precisa do comportamento externo do sistema, sem que o projetista tenha que se preocupar com detalhes de implementação;
- a definição de requisitos e decisões de projetos em diferentes níveis de abstração;

- a facilitação da comunicação entre projetista e usuário final;
- a análise da evolução da interface desde os primeiros estágios do projeto;
- a verificação de ausência de *deadlocks* e incoerências;
- a possibilidade de automação do processo.

Da mesma maneira que os modelos de tarefa, do usuário e da apresentação são partes importantes no processo de concepção de interfaces, o modelo de diálogo da aplicação mostra-se fundamental para: a descrição correta dos modos de realização das tarefas pelo usuário (o seqüenciamento da interação); o auxílio à comunicação entre o projetista e o usuário; e a descrição do fluxo de informação entre os objetos da interação (janelas, objetos de interação, etc).

Como mencionado no capítulo 2, o modelo de interação (e de arquitetura) EDITOR não possui um componente explícito de representação do diálogo. O diálogo da aplicação é representado, implicitamente, nos elementos do modelo.

3.2 - Modelo EDITOR

EDITOR é um modelo multi-agente definido em (Lula, 1992) e utilizado por (Guererro, 2002) **na metodologia MEDITE como modelo de interação e de arquitetura.**

O modelo EDITOR, como modelo conceitual de interação, adota o conceito de *edição* como paradigma central para a interação (Smith *et al.*, 1987) e inspira-se no modelo PAC (Coutaz, 1987) como modelo de arquitetura. O modelo EDITOR alia às vantagens de um modelo multi-agente (Coutaz, 1990) aos aspectos remarcáveis do modelo de Seeheim (Pfaff, 1985). PAC define a arquitetura de um sistema interativo como uma hierarquia de agentes PAC: Um agente PAC composto representa um sistema interativo, este agente, por seu turno se decompõem em agentes intermediários até os agentes PAC elementares. Todo agente PAC possui três facetas: *Apresentação(P)*, *Abstração(A)* e *Controle(C)*. Outra vantagem do componente multi-agente de PAC é permitir modelar não somente a não-sequencialidade, mas também as atividades paralelas oportunistas do usuário.

Em PAC, *Apresentação(P), Abstração(A)* e *Controle(C)* (as três facetas funcionais) são distribuídas por todos os níveis de abstração, sendo o controle distribuído e recursivo. Portanto, tal característica permite múltiplas linhas de diálogo em PAC - ou seja, dá a possibilidade de levar a cabo várias linhas de comunicação com o sistema.

PAC, portanto, é mais detalhado do que Seeheim e fornece uma arquitetura modular e distribuída. No entanto, o modelo PAC, assim como o modelo de Seeheim, está mais para um modelo de realização do que para um modelo de concepção, visto que ambos não fornecem nenhum meio ou indicação de ajuda à concepção dos seus componentes: não veiculam um método de concepção de interação nem explicitam os componentes da interface.

3.2.1 - Elementos do Modelo EDITOR

O processo de edição (paradigma central adotado pelo Modelo EDITOR para a interação) é considerado como sendo qualquer seqüência de interações entre um usuário e um sistema, onde o usuário percebe (vê, escuta, etc.) e controla (modifica, manipula, etc.) o estado da aplicação. Segundo Guerrero (Guerrero, 2002), um editor é um componente de software que suporta este processo por apresentar ao usuário uma visão particular do estado da aplicação e mecanismos para controlar e modificar esse estado.

EDITOR define a apresentação de uma interface (editor) como sendo a composição de três agentes (do tipo PAC): *Editor*, *Visão*, e *Objeto_de_Interação*, respectivamente. Cada um desses agentes apresenta as facetas *Apresentação*, *Abstração* e *Controle*:

- *Apresentação*: define a maneira como o agente pode ser percebido (associa um objeto gráfico de uma biblioteca virtual gráfica (*virtual graphical toolkit*)).
- *Abstração*: define o conceito e a função da aplicação associados ao agente (associa um objeto ou função da aplicação).
- *Controle:* define o diálogo entre as outras duas facetas e mantém a coerência entre elas.

3.2.1.1 - O Agente Editor

Um agente *Editor* define um *espaço* de visualização e manipulação de visões (agentes *Visão*) sobre a aplicação. Esta definição é traduzida por um agente PAC composto, como ilustrado na Figura (3.1) abaixo. Sua faceta *Apresentação* define esse *Espaço* através de um quadro (frame, window, janela) de uma biblioteca virtual de objetos gráficos (virtual graphical toolkit) e de uma lista de eventos associados a esse objeto (estímulos oriundos do usuário). A faceta *Abstração* permite designar um objeto

da aplicação e definir as modalidades de comunicação com ele. A faceta *Controle* gerencia a cooperação e as restrições entre os agentes *Visão* e serve de ponte aos eventos trocados entre os seus componentes *Apresentação* e *Abstração*.

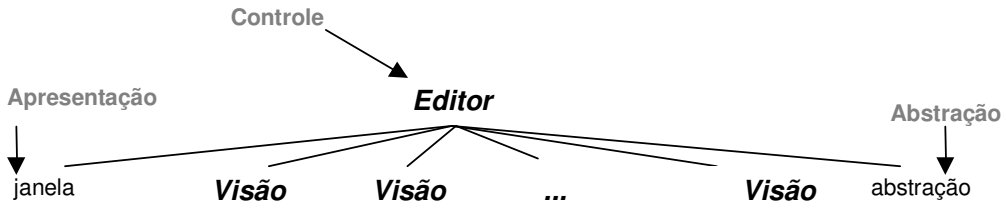


Figura (3.1): um agente *Editor* é um agente PAC composto

3.2.1.2 - O Agente Visão

Um agente *Visão* é um agente PAC composto que define o modo como é re-agrupado e visualizado um conjunto de objetos de interação (agentes *Objetos_de_Interação*). Como todo agente PAC, apresenta as três facetas. A faceta *Apresentação* define um sub-espaço através de um quadro (barra de menu, sub-janela, janela modal) e de uma lista de eventos associados a esse objeto. A faceta *Abstração* permite designar um objeto da aplicação e definir as modalidades de comunicação com ele. A faceta *Controle* define as modalidades de troca (relações) entre *Abstração* e *Apresentação* (sub-janelas), como também entre os *Objetos_de Interação.*

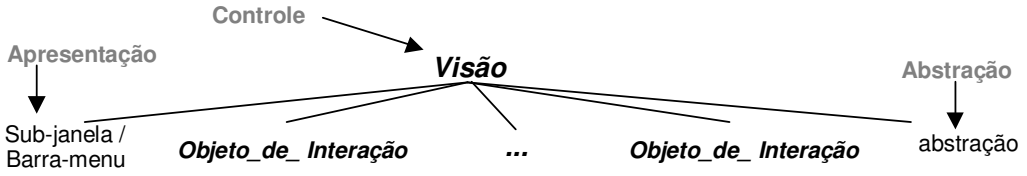


Figura (3.2): um agente *Visão* é um agente PAC composto

3.2.1.3 - O Agente Objeto_ de_ Interação

Um agente *Objeto_de_Interação* define uma forma de visualizar e manipular um objeto da aplicação a partir de objetos interativos de uma biblioteca virtual (*virtual graphical toolkit*). A faceta *Apresentação* define (i) os eventos que o *Objeto_de_Interação* é susceptível de receber provenientes do usuário (entrada) e (ii)

sua imagem através de um objeto específico de uma biblioteca virtual que lhe é associado. A faceta *Abstração* define um elemento da aplicação e as modalidades de comunicação entre eles. Enfim, a faceta *Controle* define as modalidades de trocas (de dados e de controle) e de transformação de formalismo entre os componentes.

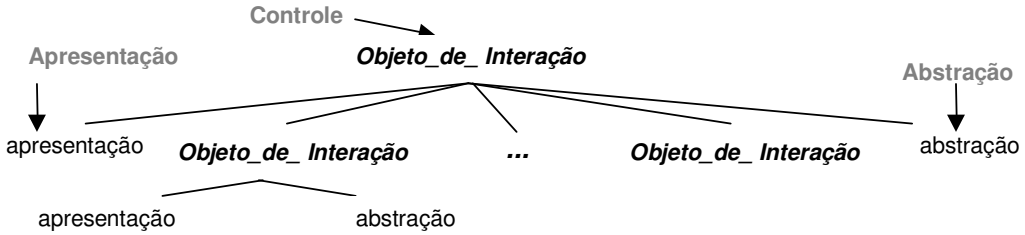


Figura (3.3): um agente *Objeto_de_Interação* pode ser simples ou composto

Conforme demonstrado acima, os agentes *Editor* e *Visão* são agentes PAC compostos e os agentes *Objetos_de_Interação* podem ser simples ou compostos. A Figura (3.4) abaixo ilustra a descrição completa da organização multi-agente de uma interface definida segundo o modelo. Esta organização multi-agente hierárquica de uma interface reflete a composição visual de sua imagem e a natureza de seus elementos (apresentação).

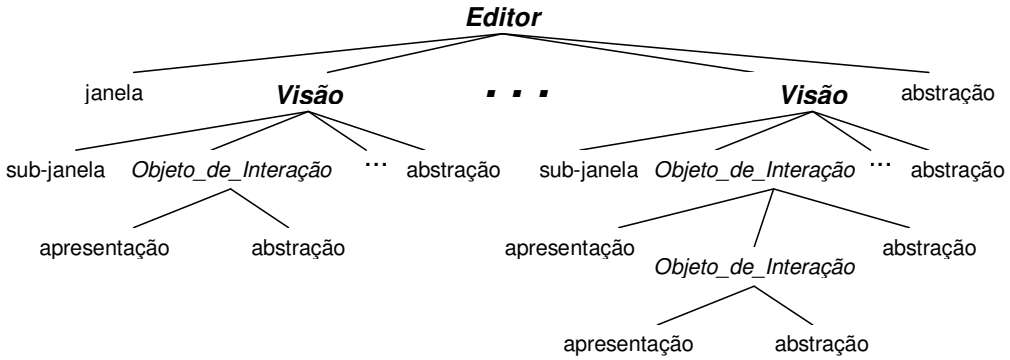


Figura (3.4): Organização multi-agente de um *Editor*

O modelo EDITOR como modelo conceitual da interação está de acordo com a Teoria da Ação (Norman, 1983) na qual o processo de realização de uma tarefa através de um sistema computacional engloba duas etapas fundamentais e que se sucedem durante a interação: avaliação e execução. Como modelo de arquitetura da interface ele reflete fielmente os conceitos e estrutura do modelo da interação e tem no paradigma de programação orientada a objeto um mecanismo natural de realização (Guerrero, 2002).

3.3 - Extensão do Modelo Editor

Como dito anteriormente, o modelo EDITOR só contempla explicitamente o componente de apresentação, sendo o diálogo da aplicação representado, implicitamente e distributivamente, através das facetas *Controle* dos elementos do modelo (*Editor*, *Visão* e *Objeto de Interação*). Isso significa que o projetista não tem uma visão geral e explícita de como o diálogo entre o usuário e o sistema é levado a efeito. Além disso, o projetista necessita se ater a detalhes de implementação, uma vez que o diálogo é descrito em termos de envio de mensagens entre objetos que implementam as facetas *Controle*. É fundamental, então, para o projetista dispor de um formalismo para a descrição explícita e rigorosa do componente de diálogo, ausente no modelo.

Suaréz (Suaréz, 2004) analisou em seu trabalho diversas metodologias de concepção de IHC com o objetivo de criar uma ontologia para unificar a grande diversidade de conceitos presentes naquelas. Um dos resultados do autor foi à concepção de três meta-modelos (tarefa, usuário e interação) que compõem a ontologia construída. O meta-modelo de interação é composto por dois componentes: *Apresentação* (modelo de apresentação) e *Diálogo* (modelo de diálogo).

<ModeloDaInteração> ::= **<Apresentação><Diálogo>**

O componente *Apresentação* define a apresentação da interface através dos elementos *Espaço*, *Visão* e *Objeto de Interação.* Fazendo um paralelo entre o modelo EDITOR (Lula, 1992) e o meta-modelo apresentado por Suaréz (Suaréz, 2004), um *Agente Editor* (modelo EDITOR), equivale a um *Espaço* (meta-modelo de interação). Um *Agente Visão* (modelo EDITOR) corresponde a uma *Visão* (meta-modelo de interação). E um *Agente Objeto de Interação* (modelo EDITOR) equivale a um *Objeto de Interação.*

O componente *Apresentação* é descrito conforme a gramática apresentada a seguir, extraída de (Lula, 1992).

<editor> ::= **quadro** <visão-barra> {<visões>} **abstração** |
quadro {<visões>} **abstração** |
quadro-fixo {<visões>} **abstração**

<visão-barra**>** ::= **barra-de-menu** {<objeto-de-interação-menu-barra>} **abstração**

<visão> ::= **sub-janela-gráfica** {<objeto-de-interação-gráfico>} **abstração** |
sub-janela-controle {<objeto-de-interação-controle>} **abstração** |
sub-janela-texto abstração

<objeto-de-interação-menu-barra> ::= **menu-barra abstração**

<objeto-de-interação-gráfico> ::= <objeto-de-interação-simples> |
<objeto-de-interação-grupo > |
<objeto-de-interação-grafo> |
<objeto-de-interação-composto> |

<objeto-de-interação-controle> ::= <objeto-de-interação-comando> |
<objeto-de-interação-mensagem> |
<objeto-de-interação-texto-editável> |
<objeto-de-interação-caixa-de-opções> |
<objeto-de-interação-menu>

<objeto-de-interação-simples> ::= <objeto-de-interação-ícone> |
<objeto-de-interação-imagem> |
<objeto-de-interação-figura> |
<objeto-de-interação-cadeia-de-caracteres>

<objeto-de-interação-ícone> ::= **ícone-a abstração** |
.....
ícone-n abstração

<objeto-de-interação-imagem> ::= **imagem-a abstração** |
.....
imagem-m abstração

<objeto-de-interação-figura> ::= **retângulo abstração** |
.....
termômetro abstração

<objeto-de-interação-cadeia-de-caracteres> ::= **cadeia-de-caracteres abstração**

<objeto-de-interação-composto> ::= **composto** {<objeto-de-interação-composto>} **abstração** |
<objeto-de-interação-simples> |
<objeto-de-interação-grupo>

<objeto-de-interação-grupo> ::= **grupo abstração**
<objeto-de-interação-comando> ::= <objeto-de-interação-botão-comando> |
<objeto-de-interação-ícone-comando>

<objeto-de-interação-botão-comando> ::= **botão-comando abstração**

<objeto-de-interação-ícone-comando> ::= **ícone-comando abstração**

<objeto-de-interação-mensagem> ::= <objeto-de-interação-mensagem-texto> |
<objeto-de-interação-mensagem-ícone>

<objeto-de-interação-mensagem-texto> ::= **mensagem-texto abstração**

<objeto-de-interação-mensagem-ícone> ::= **mensagem-ícone abstração**
<objeto-de-interação-texto-editável> ::= **texto-editável abstração**

<objeto-de-interação-caixa-de-opções> ::= **caixa** {<objeto-de-interação-opção-única>}
abstração |
caixa {<objeto-de-interação-opções-múltiplas>}
abstração

<objeto-de-interação-opção-única> ::= **opção-única abstração**

<objeto-de-interação-opção-múltopla> ::= **opção-múltipla abstração**

<objeto-de-interação-menu> ::= <objeto-de-interação-menu-permanente> |
<objeto-de-interação-menu-de-rolagem>

<objeto-de-interação-menu-permanente> ::= **menu-permanente abstração**
<objeto-de-interação-menu-de-rolagem> ::= **menu-de-rolagem abstração**

Segundo Suárez (2004), o componente *Diálogo* descreve o fluxo da interação (seqüências de trocas de mensagens) entre o usuário e a aplicação e é comumente descrito na forma de um diagrama de transição de estados (Brochot, 1990). Então, o componente *Diálogo* pode ser descrito de maneira análoga a uma máquina de estados finitos, onde os estados representam os elementos de apresentação (espaços, visões e objetos de interação) e os arcos representam os estímulos empregados no processo de comunicação. Deste modo, o componente *Diálogo* é composto por uma *lista* de elementos do tipo **Estado**, uma *lista* de elementos do tipo **Estímulo**, a descrição do **EstadoInicial**, uma *lista* de **EstadosFinais** contendo a descrição de elementos do tipo **Estado** e a descrição da **FunçãoDeTransição,** conforme a gramática apresentada a seguir:

<Diálogo> **::=** **{<Estado>}**
{<Estímulo>}
<EstadoInicial>
{<EstadoFinal>}
<FunçãoDeTransição>

Um **Estado** é composto por um *Identificador* e por um *LinkEspaço* ou um *LinkVisão* ou um *LinkObjetoDeInteração*.

<Estado> **::=** *Identificador*
LinkEspaço **|** *LinkVisão* **|** *LinkObjetoDeInteração*

Já um **Estímulo** representa um conjunto de símbolos terminais, ou seja, é um elemento que compõe o alfabeto do autômato.

<Estímulo> **::=** *Terminais (Alfabeto do autômato)*

Um **EstadoInicial** é composto por um *Identificador* e um *LinkEspaço* que referencia o ponto de partida do **Diálogo** da interação, enquanto que um **EstadoFinal** é

composto por um *Identificador* e um *LinkObjetoDeInteração* que referencia um **Estado** capaz de caracterizar o reconhecimento do diálogo da interação.

<EstadoInicial>	**::=**	*Identificador* *LinkEspaço*
<EstadoFinal>	**::=**	*Identificador* *LinkObjetoDeInteração*

Já a **FunçãoDeTransição** é composta por um *Identificador* seguido de uma *lista* de elementos que caracterizam a passagem da origem (de um *LinkObjetoDeInteração* ou um *LinkVisão* ou um *LinkEspaço*, seguido de uma *lista* de elementos do tipo **EfeitoDeInteração**) a um ou mais destinos, com vistas ao alcance de um outro **Estado** do autômato.

<FunçãoDeTransição>	**::=**	*Identificador* {*LinkObjetoDeInteração***{<EfeitoDeInteração>}**} {*LinkVisão***{<EfeitoDeInteração>}**} {*LinkEspaço***{<EfeitoDeInteração>}**}

Por fim, um **EfeitoDeInteração** é composto por um *Identificador* seguido de uma relação de *associação* entre um *estímulo* e um *link*, quer seja um *LinkObjetoDeInteração*, um *LinkVisão* ou um *LinkEspaço*.

<EfeitoDeInteração>	::=	*Identificador* *Associação(Estímulo,LinkObjetoDeInteração)* | *Associação(Estímulo,LinkVisão)* | *Associação(Estímulo,LinkEspaço)*

Assim, o modelo de interação adotado doravante em MEDITE será o modelo EDITOR Estendido**,** composto pelos componentes *Apresentação*, cuja definição encontra-se em (Lula, 1992), acrescido do um componente *Diálogo*, assim como proposto por Suárez em (Suárez, 2004).

3.4 - Conclusão

Realizada a extensão do modelo EDITOR, a próxima tarefa a ser empreendida neste trabalho será a escolha de uma representação (formalismo) apropriada para o modelo EDITOR Estendido. No próximo capítulo, serão apresentados as características e os requisitos para a escolha de formalismos de representação do diálogo da interação,

assim como será um apresentado um estudo acerca dos formalismos “candidatos” à representação do componente de diálogo do modelo EDITOR Estendido.

Capítulo 4 – Representação do Componente *Diálogo* do Modelo EDITOR Estendido

Este capítulo refere-se à escolha de uma representação (formalismo) apropriada para o modelo EDITOR Estendido. Inicialmente, serão apresentados as características e os requisitos para a escolha de formalismos de representação do diálogo da interação. Em seguida, serão levantadas algumas considerações acerca dos formalismos "candidatos" à representação do componente de diálogo do modelo EDITOR. Após tais considerações, será apresentado o formalismo escolhido para a representação do componente *Diálogo* do modelo EDITOR Estendido e, por fim, o processo de obtenção do componente *Diálogo* a partir do modelo da tarefa e do componente de apresentação do modelo EDITOR Estendido.

4.1 - Características e Requisitos para Formalismos de Representação do Diálogo da Interação.

Para Elwert *et at.* (Elwert e Schlungbaum, 1995 ; Elwert *et al., 1996*) um formalismo representativo para a modelagem do diálogo da interação deve satisfazer os seguintes requisitos:

- a especificação do diálogo deve ser de fácil elaboração e entendimento;
- a especificação deve ser precisa. O comportamento do sistema deve ser claro para cada estímulo de entrada possível;
- a verificação de consistência e de ausência de (conflitos) *deadlocks* na especificação do diálogo deve ser fácil de ser realizada;
- o formalismo de especificação deve ser poderoso o suficiente para expressar comportamentos não triviais de sistemas com o mínimo de complexidade (concorrência e diálogos modais);
- a descrição do comportamento do sistema deve estar desacoplada das características de implementação. Deve haver, portanto, uma separação clara entre funcionalidades e a implementação das mesmas;

- o formalismo deve possibilitar a construção de protótipos da interface do sistema diretamente da especificação conceitual da interface.

Além desses requisitos definidos por Elwert (Elwert *et al., 1996*), os formalismos de representação do componente de diálogo da interação podem ser visuais ou gráficos. Tse e Pong (Tse e Pong, 1991) consideram que uma representação gráfica é mais compreensível do que uma representação textual. Para os autores, isso se dá pelos seguintes motivos: uma representação gráfica mostra naturalmente a hierarquia e concorrência entre elementos da interação; possibilita a leitura com mais seletividade ao invés da leitura linear; possibilita a redução do número de conceitos manipulados na memória de curto termo; o projetista pode passar naturalmente de níveis mais altos de detalhamento para níveis mais baixos. Entretanto, Harel (Harel, 1988) sugere que estes formalismos visuais (gráficos) devem ser utilizados com cautela. O esforço para aprender e compreender o formalismo não pode ser maior do que a sua proposta de facilitação ao entendimento do projeto. Um outro requisito desejável e não considerado pelos autores supracitados seria a existência ou possibilidade de existência de auxílio computacional para a representação e edição do formalismo.

Portanto, para representar o componente *Diálogo* do modelo EDITOR Estendido, propõe-se neste trabalho um formalismo que satisfaça os requisitos definidos por Elwert, possua uma representação gráfica e que possa prover um suporte ferramental.

4.2 - Formalismos para Representação do Componente de Diálogo da Interação.

Como visto no capítulo 3, o componente *Diálogo* do modelo EDITOR Estendido descreve o fluxo da interação (seqüências de trocas de mensagens) entre o usuário e a aplicação e pode ser descrito na forma de um diagrama de transição de estados (Brochot, 1990). Diversos são os formalismos com abordagem estados/transição passíveis de representação do componente de diálogo de uma aplicação, entre os quais, os mais representativos e mais referenciados na literatura são: Autômatos-Finitos (Douglass, 2004 ; Cassandra e Lafortune, 1999; LFA, 2004); Redes de Petri (e suas variações) (Figueiredo e Braga, 1997; Jensen, 1992 ; Jensen, 1995; Guerrero, 2002b); UAN (Hix e Hartson, 1988); CSP (Ward e Mellor, 1985); Dialogue Graphs (Elwert *et al.*, 1996); Windows Transition Notation (Vanderdonckt *et al.*, 2003) e Statecharts (Harel, 1987; Rumbaugh et al., 1998; Eshuis e Wieringa, 2000; Fowler e Scott, 2000; UML v1.5, 2004; Douglass, 2004).

Os formalismos CSP e UAN, embora sejam baseados em eventos/transição, não apresentam uma representação gráfica e, portanto, não serão levados em consideração na escolha do formalismo para a extensão do modelo EDITOR. Os Autômatos Finitos, por seu turno, não permitem representar informações relativas à hierarquia e à concorrência de estados. As redes de Petri e suas variações, em que pese o seu atendimento à maioria dos requisitos exigidos acima, apresentam um grande inconveniente, qual seja a dificuldade de entendimento de sua representação, que exige um esforço adicional por parte dos envolvidos no processo de concepção de interfaces. Dialogue Graphs também não será considerado visto que é uma extensão do formalismo redes de Petri. Outro formalismo também analisado e que *a priori* atende aos requisitos supracitados para a representação do diálogo da interação do sistema é o formalismo Windows Transition Notation (Vanderdonckt *et al.*, 2003). Entretanto, o WTN só contempla o diálogo interjanelas.

Dos formalismos citados supracitados, o que satisfaz minimamente todos os requisitos levantados é o formalismo Statechart, que será apresentado a seguir.

4.3 - Statecharts

Os Statecharts foram propostos por Harel (Harel, 1987) para superar os problemas encontrados nas máquinas de estados e diagramas de seqüência convencionais. Os Statecharts são um aperfeiçoamento das máquinas de estados finitos (ou autômatos de estados finitos) tradicionais. Statecharts podem ser representados formalmente, através de uma notação matemática, ou ainda pela sua notação gráfica.

Os conceitos de hierarquia, concorrência e atividades, permitem ao projetista a modelagem de sistemas complexos. Ajudam também a diminuir o tamanho da modelagem desses sistemas, comparado-se com as máquinas de estados convencionais, uma vez que, segundo Harel, os Statecharts não sofrem com o problema do crescimento exponencial de estados.

Basicamente os elementos principais dos Statecharts são: Estados, Transições e Ações. Os *Estados* são condições de existência distinguíveis que persistem por um período de tempo significante (Douglass, 2004). *Transições* são a maneira como os objetos mudam de estados em resposta a um evento. *Ações* são comportamentos atômicos, ou operações, processados em uma máquina de estados quando se entra ou se sai de um evento.

O formalismo Statechart atende aos requistos levantados para a representação do componente de diálogo do modelo EDITOR Estendido. O Statechart é um formalismo que conta com uma representação gráfica, simples, precisa, fácil de entender e de utilizar, capaz de representar comportamentos não triviais do sistema (concorrência entre os elementos do sistema e comportamentos modais), permite a construção de protótipos diretamente de sua especificação formal e ainda possui uma série de ferramentas (para sua edição e representação) disponíveis. Ferramentas que tenham suporte à linguagem XMI (XML Metadata Interchange) (XMI, 2004), podem ser utilizadas como interface gráfica a qualquer módulo funcional que implemente o formalismo Statechart, sendo uma alternativa à criação de uma interface gráfica para tal módulo. O formalismo Statecharts faz parte da definição da Linguagem de Modelagem Unificada - UML (Unified Modeling Language) (Rumbaugh et al., 1998; Fowler e Scott, 2000; UML v1.5, 2004) que vem se tornando uma linguagem padrão para a modelagem e representação de sistemas computacionais. Esse fato evidência fortemente que a utilização dos Statecharts facilitará a comunicação entre os envolvidos no processo de concepção de interfaces, sendo, portanto, mais um dos motivos para a escolha dos Statecharts como formalismo de representação do componente de diálogo para a extensão do modelo de interação EDITOR.

4.3.1 - Elementos do Statechart

Apesar de possuir uma representação matemática, para o estudo proposto nesse trabalho, a notação gráfica é a mais importante.

Os estados de um Statechart podem ser classificados como estados or-states ou and-states. Nos estados or-states, apenas um estado do Statechart pode estar ativo em um dado momento de tempo. Nos estados and-states, vários estados do Statechart podem estar ativos ao mesmo tempo, ou seja, um estado ativo em cada região ortogonal (falaremos mais em regiões ortogonais mais adiante). Esta característica dos estados do tipo and-state permite a concorrência pelo formalismo.

Para um melhor entendimento, serão apresentadas inicialmente as características dos Statecharts com estados tipo or-state.

4.3.1.1 - Estados

Os estados de um Statechart tipo or-state são representados por um retângulo com os cantos arredondados. Eles podem ser simples ou compostos. Um estado simples não contém nenhum sub-estado. Nos estados compostos, cada estado pode conter mais de um sub-estado. Observem a Figura (4.1) o estado S0, contém dois sub-estados, S0_1 e S0_2. Se o estado S0 está ativo, significa dizer que o objeto estará em S0_1 ou S0_2. Uma vantagem dos estados compostos está na noção de hierarquia, representada por estados aninhados. Esse tipo de representação - hierárquica - torna o formalismo bastante poderoso para a modelagem de sistemas complexos. Ao contrário das máquinas de estados convêncionais, como dito anteriomente, não há um aumento exponêncial do número de estados para cada novo estado que necessite ser representado.

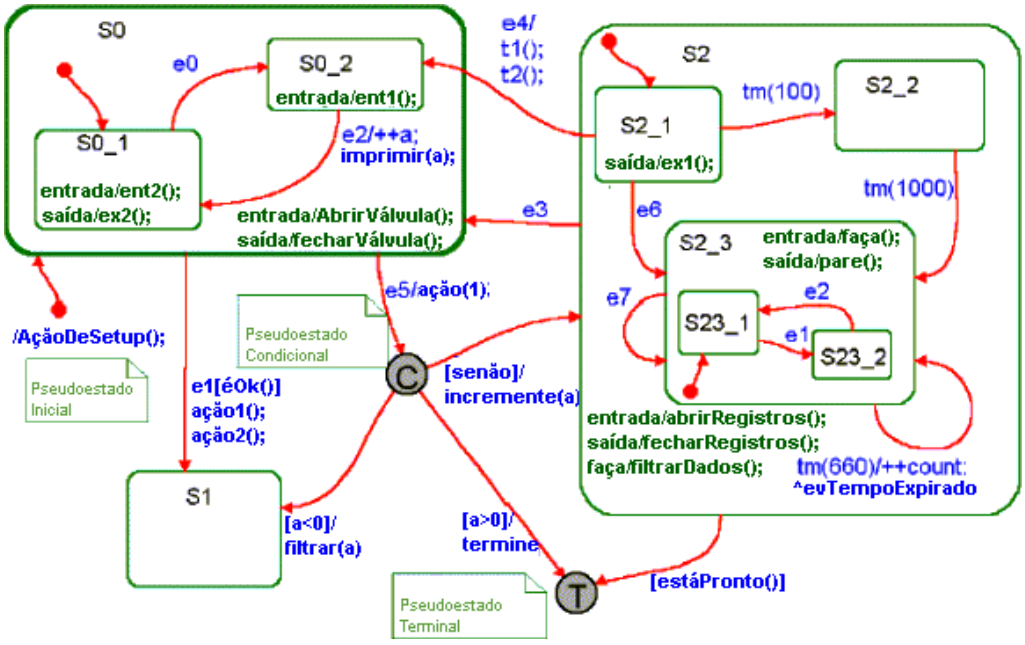


Figura (4.1): Exemplo de um Statechart do com estados tipo or-state.

Existem também outros tipos de estados, chamados de pseudo-estados. São eles: inicial, final, condicional e histórico[1]. O pseudo-estado inicial (representado com uma seta com uma circunferência na sua estremidade superior), como o próprio nome sugere, indica qual será o estado de início do Statechart, assim como qual será o sub-estado

[1] Existem ainda os pseudo-estados join, fork e junction que não são necessários aos propósitos desse trabalho.

inicial de cada estado composto. Observem a Figura (4.1), o pseudo-estado inicial do Statechart leva ao estado S0. Notem também que cada estado composto tem um pseudo-estado inicial indicando onde iniciar-se-á a ativação, daquele estado, caso seu superestado seja atingido. O pseudo-estado condicional (representado na Figura acima por um "C" circunscito ou opcionalmente por um losângulo) avalia uma condição lógica - oriunda do guarda dos eventos (falaremos mais sobre os guardas adiante) - e decide para qual estado será realizada a transição. Ou seja, o pseudo estado condicional é uma abreviação de múltiplas transições de um mesmo evento, em um estado de origem, para diferentes estados de destino. Todos esses eventos são "guardados" por uma condição lógica (guarda) que resolverá qual será o estado destino. O pseudo-estado histórico (representado por um "H" circunscrito) serve como uma referência em caso de transições entre estados. Se um superestado contiver um pseudo-estado histórico, isso significa dizer que quando esse superestado for deixado e quando houver uma nova transição para ele novamente, este estado retornará ao seu último ponto ativo (se o estado tiver sido visitado anteriormente). Por fim, o pseudo-estado terminal indica que a máquina de estados chegou ao ponto final de sua execução.

4.3.1.2 - Transições

As transições são representadas nos Statecharts por setas (arcos) que ligam o estado origem ao estado destino. As transições são realizadas através de eventos gerados no/ao sistema. Observe-se a Figura (4.1) o estado S0, contém dois sub-estados, S0_1 e S0_2. como acima mencionado, se o estado S0 está ativo, significa dizer que o objeto estará em S0_1 ou S0_2. Os sub-estados de S0 (S0_1 e S0_2) respondem aos eventos de S0 (e1 e e5) e também aos eventos deles provinêntes (S0_1 responde ao evento e0 e S0_2 responde ao evento e1). Note-se que o evento e1 (do estado s0), conta com um guarda [éOk()]. Os guardas são expressões ou funções lógicas que retornam valor verdadeiro ou falso. Se o estado S0 estiver ativo e o evento e1 for gerado, haverá uma transição se e somente se o guarda [éOk()] for avaliado como verdadeiro, caso contrário, o evento é descartado e o objeto continua no estado S0. Portanto, os eventos são essenciais para haver as transições de estados.

Os eventos podem ser de 4 tipos: Signal Event, Call Event, Time Event e Change Event. Signal Events são eventos relativos a um processo externo assíncrono, ou seja, são os eventos provocados por agentes externos ao sistema, sendo, portanto, os mais

corriqueiros nos Statecharts. Time Events são eventos gerados em um certo período de tempo, por exemplo, a cada 100 milisegundos. A Figura (4.1) apresenta 3 eventos do tipo Time Event, por exemplo, o evento tm(100) no estado S2_1. Um Call Event é um evento proviniente de uma execução de um procedimento de um objeto, e um Change Event é um evento proviniente de uma mudança de um atributo de um objeto.

A nomenclatura geral para uma transição é:

Nome-do-evento '('lista-de-parâmetros')' '['guarda']' '/' lista-de-ações '^' lista-de-eventos

Todos esses elementos são opcionais, inclusive o nome do evento, isso quer dizer que podem haver eventos sem um nome, chamados de eventos não nomeados.

4.3.1.3 - Ações e Atividades.

Ações e atividades são outra característica dos Statecharts que não estão presentes nos autômatos-finitos. Ações são pequenos comportamentos (procedimentos) atômicos executados em certos pontos dessas máquinas de estados. As ações demandam um tempo de realização próximo a zero. As ações podem ser realizadas no momento em que um evento é disparado (após o '/' na assinatura do evento), ao entrar em um estado (após o termo "entrada/" representado no estado) ou ao sair de um estado (após o termo "saída/" representado no estado). A ordem de realização das ações é sempre a seguinte: primeiramente as ações de saída do estado de origem, em seguida as ações do evento (de transição) e por fim as ações de entrada do estado de destino. Caso a transição originasse-se de sub-estados de um estado composto, começar-se-ia sempre dos sub-estados mais internos (mais profundos) até o super estado. Observem na Figura (4.1) o sub-estado S23_1, caso este estado estivesse ativado e o evento e3 fosse disparado, a ordem de realização das ações seria a seguinte: ação de saída de S23_1 (se houvesse), ação de saída de S2_3 ("saída/pare();"), ação de saída de S3 ("saída/fecharRegistros();"), a ação do evento e3 (se houvesse) e, por fim, a ação do estado de destino S0 ("entrada/abrirVálvula();").

As atividades são procedimentos que demandam um maior tempo. São interrompíveis e realizadas em quanto o estado está ativo. São iniciadas após a realização das ações de entrada e terminadas quando o estado fica inativo (após as ações de saída).

Elas são representadas após o termo “do/” no Statechart. A Figura (4.1) apresenta uma atividade (“faça/filtrarDados();”) no estado S2.

4.3.1.4 - Estados And-State

Os estados and-state são chamados também de regiões ortogonais. Diferentemente dos estados or-state, onde apenas um estado pode estar ativo em um dado momento, nos estados and-state cada uma das regiões ortogonais pode ter um estado ativo. Observem a Figura (4.2) a existência de um estado chamado “working”. O estado “working” contém três regiões separadas por linhas tracejadas. Cada uma dessas regiões poderá ter um estado ativo ao mesmo tempo. Essa característica dos estados do tipo and-states proporciona a concorrência entre estados (impossível de ser representada com a utilização de autômatos-finitos). Mais ainda, este tipo de estado permite modelar diversos tipos de aspectos diferentes que podem estar ativos em um determinado objeto. Todas as demais características relativas aos estado or-state são válidas para este tipo de espaço.

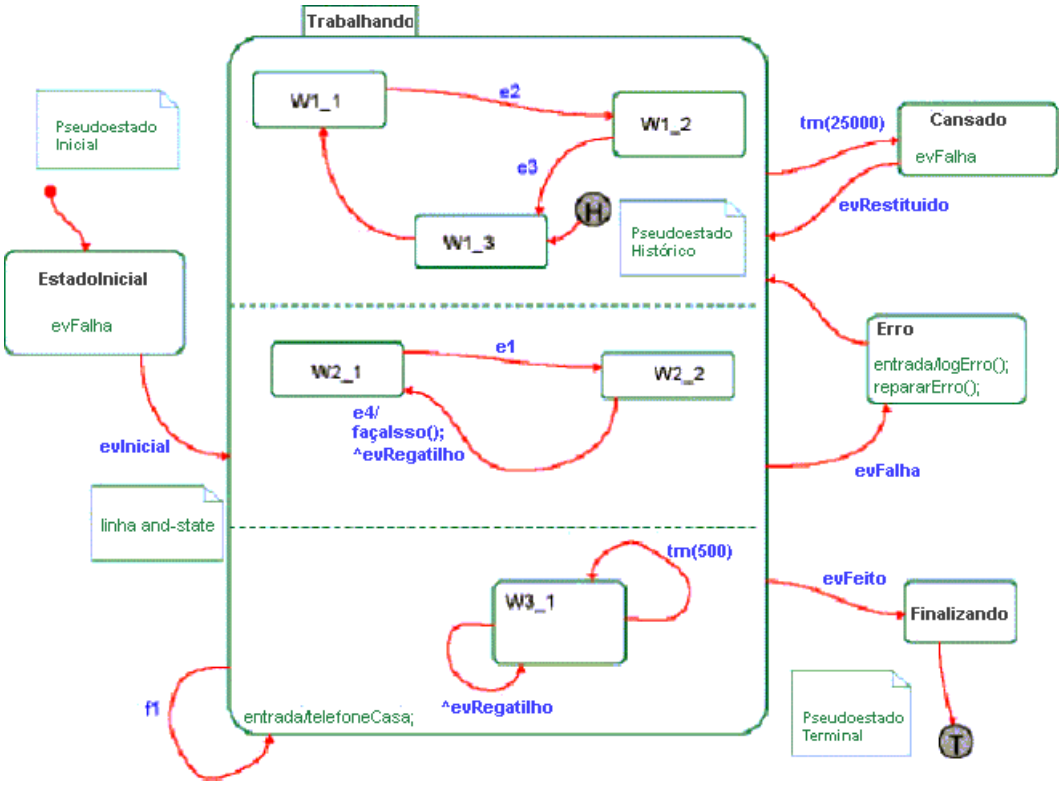


Figura (4.2): Statechart com estado and-state

4.4 - Representação Statechart do Componente *Diálogo*.

O componente *Diálogo* do modelo EDITOR Estendido é obtido de elementos do modelo da tarefa (TAOS) e do componente *Apresentação* do modelo EDITOR. A partir do componente de apresentação, identificam-se os estados que comporão o Statechart. Os eventos e as transições possíveis são adquiridos do modelo da tarefa através de alguns de seus atributos como será apresentado nas seções seguintes.

4.4.1 – Identificação dos Estados do Statechart.

Os *estados* do Statechart são obtidos através dos elementos do componente *Apresentação* do modelo da interação EDITOR Estendido. O modelo da interação, como fora anteriormente visto, está representado sob a forma de árvores EDITOR, portanto, de cada árvore será obtido um pequeno trecho do Statechart de representação da interface do sistema. O elemento *EDITOR* será associado a um *estado composto* (super estado). Cada elemento *Visão* será, também, associado a um *estado composto*. Este *estado composto* (associado ao elemento *Visão*) será um sub-estado do estado associado ao elemento *EDITOR*. Cada elemento *Objeto de Interação* será associado a um *estado simples*, sendo este um sub-estado do *estado composto* associado ao elemento *Visão*. A Figura (4.3) ilustra essa situação. O estado maior (laranja) representa um *EDITOR* composto por dois sub-estados. Estes sub-estados representam as *Visões* (beges), que são compostos por *estados simples* (verdes), representando os *Objetos de Interação.*

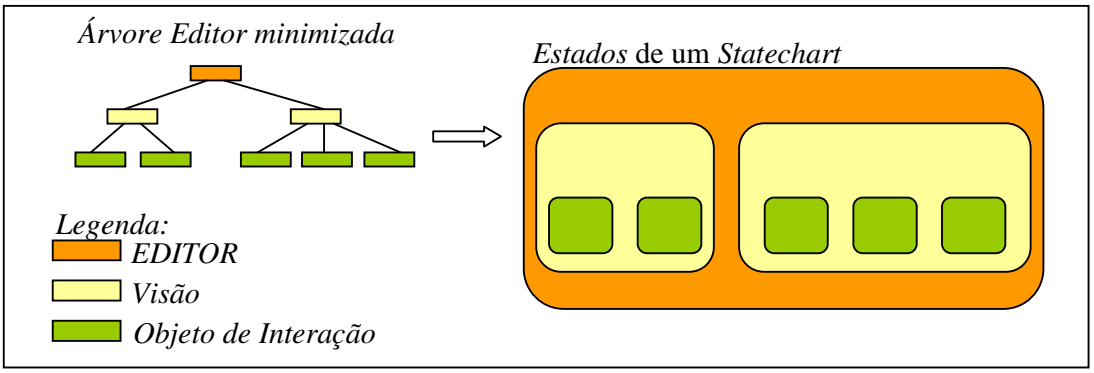


Figura (4.3): Obtenção dos *estados* de um *Statechart* a partir do modelo de interação.

4.4.2 - Identificação dos Eventos e Transições do Statechart.

As modificações propostas nesse trabalho, dizem respeito à passagem da primeira para a segunda etapa da metodologia MEDITE (maiores detalhes serão vistos no capítulo 5). Na segunda etapa da metodologia, qual seja, obtenção da especificação parcial do modelo da interação, ainda não existe o conceito de objetos de interação concretos. Nesta etapa – assim como em diversas outras metodologias, como em TRIDENT e ADEPT[2], por exemplo – trabalha-se com objetos de interação abstratos ou categorias de objetos (objetos de controle, de apresentação, entre outros).

Dessa maneira, os eventos que irão disparar as transições são eventos gerais, tais como, eventos de *realização* ou *não realização* de uma condição. Portanto, não se trabalha com eventos relativos a objetos de interação concretos como botões, caixas de textos, ícones, etc. No momento da especificação completa do modelo da interação (próxima etapa da metodologia), o tipo de cada objeto será escolhido, ou seja, haverá a passagem de objetos abstratos para objetos concretos. Neste momento os eventos deverão ser estendidos para cada tipo de objeto selecionado de uma *toolkit* adotada. Esta etapa, entretanto, não faz parte do escopo deste trabalho.

De posse dessas restrições, os eventos possíveis no modelo são:

Quadro (4.1): Eventos aceitos pelo sistema.

	Elementos do Modelo da Interação		
	Objetos de Interação	**Visão**	**EDITOR (Espaço)**
Eventos	eProcedimentoRealizado - ePR eProcedimentoNãoRealizado – ePNR eRepetirProcedimento – eRP eRealizarLink - eRL	eVisãoNãoRealizada – eVNR eRepetirVisão – eRV	eEspaçoNãoRealizado - eENR eRepetirEspaço – eRE

Os eventos ePNR, eVNR e eENR indicam que a tarefa associada a um *Objeto de Interação*, uma *Visão* ou um *Editor* (Espaço), respectivamente, não foi realizada, e portanto, tais eventos serão sempre acompanhados de uma ação que cancela todas as ações e atividades do estado de origem. Desta maneira, toda vez que os evento ePNR, eVNR e eENR forem gerados suas assinaturas serão acompanhadas de uma cláusula: <evento> **/ AnulaAçõesEAtividadesDaOrigem();**. O evento eRealizarLink (eRL) só é admissível a um objeto de interação, se o mesmo, for responsável pela realização de um

[2] Trident e Adept utilizam o conceito de Objetos de Interação Abstratos e Objetos de Interação Concretos.

link a um outro *Editor*. Quando um *Objeto de Interação* fizer link a um *Espaço* Adjacente (diálogo inter-Editores) o fará através do evento eRealizarLink (eRL).

Os eventos, possíveis a cada tipo de elemento do modelo EDITOR Estendido são obtidos através do atributo *ocorrência* da tarefa associada a cada elemento. A *ocorrência* (0,0) implica dizer que a tarefa não deve ser realizada. A *ocorrência* (0,1) implica dizer que a tarefa pode não ser realizada (tarefa opcional) ou ser realizada apenas uma vez. A *ocorrência* (0,n) implica dizer que a tarefa pode não ser realizada (tarefa opcional) ou ser realizada n vezes. A *ocorrência* (1,1) implica dizer que a tarefa tem que ser realizada uma vez. Por fim, a *ocorrência* (1,n) implica dizer que a tarefa pode ser realizada apenas uma vez ou n vezes. O Quadro (4.2) ilustra a combinação ocorrência *versus* elemento da interação.

Quadro (4.2): Combinação elementos da interação x ocorrência da tarefa associada.

		Elemento da Interação		
		Objetos de Interação	**Visão**	**EDITOR (Espaço)**
Ocorrência da tarefa associada	**(0,0)**	eProcedimentoNãoRealizado – ePNR	eVisãoNãoRealizada – eVNR	EespaçoNãoRealizado – eENR
	(0,1)	eProcedimentoNãoRealizado – ePNR eProcedimentoRealizado – ePR eRealizarLink - eRL	eVisãoNãoRealizada – eVNR	eEspaçoNãoRealizado – eENR
	(0,n)	eProcedimentoNãoRealizado – ePNR eProcedimentoRealizado – ePR eRepetirProcedimento – eRP eRealizarLink - eRL	eVisãoNãoRealizada – eVNR eRepetirVisão – eRV	eEspaçoNãoRealizado – eENR eRepetirEspaço – eRE
	(1,1)	eProcedimentoRealizado – ePR eRealizarLink - eRL	-	-
	(1,n)	eProcedimentoRealizado – ePR eRepetirProcedimento – eRP eRealizarLink - eRL	eRepetirVisão – eRV	eRepetirEspaço – eRE

As transições entre estados do Statechart podem ser: entre estados internos a um super estado (*estado composto*), ou seja, o diálogo intra-*Editor*; ou entre estados internos a um super estado "A" para um outro super estado "B", sendo este o diálogo inter-*Editor*es.

O diálogo intra-*Editor* é obtido através dos atributos *método* das tarefas associadas a cada um dos elementos da árvore EDITOR. O valor do atributo *método* é uma expressão formada utilizando-se os operadores *SEQ*, *OR*, *XOR*, *AND*, *SIM* e *PAR*, conforme (Cordeiro, 2003), que representam quais as seqüências, restrições e condições de realização de uma tarefa, implicando nas mesmas condições de realização (transição)

dos estados do Statechart. Utilizar o operador *SEQ* significa dizer que as tarefas têm que ser realizadas da em seqüência, da esquerda para a direita. Com o operador *OR*, as tarefas podem ser ou não realizadas (depende da *ocorrência*) e em qualquer ordem. Com o operador *XOR*, apenas uma das tarefas pode ser realizada. Com o operador *AND*, todas as tarefas tem que ser realizadas, não importando a ordem de realização. Com o operador *SIM*, as tarefas são realizadas simultaneamente, ou seja, quando uma começa as demais começam e quando uma termina as demais terminam. Com o operador *PAR*, as tarefas são realizadas paralelamente, ou seja, concorrentemente, sem a necessidade de coincidência de começo e fim.

Se a expressão de um método possuir mais de um operador, os operadores mais internos terão no Statechart um estado que o represente (a Figura (4.6) seção 4.5 representa esta situação). Caso haja apenas um operador, não há necessidade da inclusão de mais um estado no Statechart. Embora tal fato não esteja diretamente relacionado ao encadeamento das transições, o fato está ligado à presença de múltiplos operadores para uma mesma tarefa.

A combinação entre os atributos *ocorrência* (que nos fornece os eventos possíveis a cada um dos elementos do modelo EDITOR) e *método* (seqüência, restrições e condições das transições) nos dá todas as transações internas possíveis ao super estado de um Statechart, ou seja, transações intra-EDITOR. O Quadro (4.3) representa todas as combinações *método* x *ocorrência* possíveis.

Quadro (4.3): Combinações possíveis *ocorrência* versus *método*.

		Método de Realização das Sub-tarefas.					
		SEQ	**XOR**	**OR**	**AND**	**SIM**	**PAR**
Ocorrência	**(0,0)**	-	-	-	-	-	-
	(0,1)	X	X	X	-	X	X
	(0,n)	X	X	X	-	X	X
	(1,1)	X	X	X	X	X	X
	(1,n)	X	X	X	X	X	X

O diálogo inter-*Editores*, por seu turno, é obtido através da hierarquia das árvores EDITOR. Uma árvore EDITOR é dita hierarquicamente superior a outra, se a tarefa associada ao seu elemento *Editor* estiver em um nível mais alto do que a tarefa

associada ao elemento *Editor* da outra árvore. Assim, existem dois tipos de diálogos inter-*Editores* possíveis: (i) de uma árvore hierarquicamente superior para uma árvore hierarquicamente inferior; e (ii) de uma árvore hierarquicamente inferior para uma árvore hierarquicamente superior. No primeiro caso, a transição será representada no Statechart através do evento Realizar Link (eRL) do objeto de interação responsável pelo link entre as árvores. Já no segundo, a transição será representada no Statechart através do evento Procedimento Realizado (ePR) do objeto de interação responsável pelo link entre as árvores.

Apresenta-se a seguir, através de exemplos, o processo de aquisição dos Statecharts a partir da árvore da tarefa e das árvores EDITOR. Para maiores detalhes sobre todas as possíveis seqüências, condições e restrições necessárias relativas aos operadores da expressão valor do atributo *método*, assim como todas as possíveis combinações do tipo *método* x *ocorrência,* o leitor deve consultar o **Anexo A**.

4.5 - Exemplo 1 – Diálogo Intra-*Editor*

A Figura (4.4) apresenta a árvore EDITOR A. Tal árvore foi obtida através do modelo da tarefa, sendo o trecho da árvore da tarefa apresentada pela Figura (4.5).

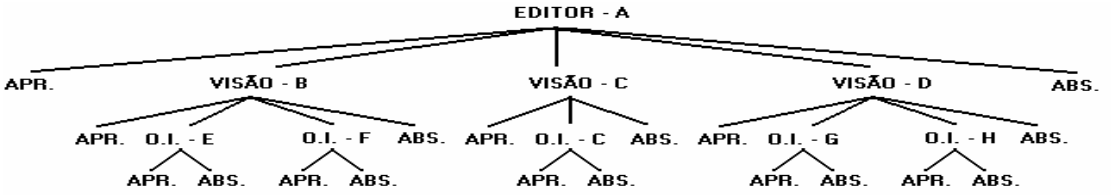


Figura (4.4): Árvore EDITOR A

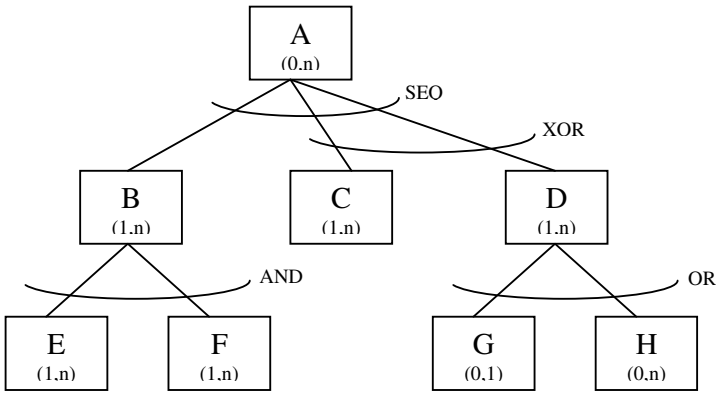


Figura (4.5): Trecho de Árvore da Tarefa correspondente à árvore EDITOR A.

Como explicado na seção anterior, a cada elemento da árvore EDITOR, será associado um *estado* no Statechart. Os eventos são retirados das ocorrências e as transições (que dão a seqüência das mudanças de estado) são obtidas através da expressão valor do atributo *método*. Dessa maneira, o Statechart correspondente a esse trecho de árvore é representado a seguir pela Figura (4.6).

Pode-se observar que a tarefa A é realizada da seguinte maneira: *SEQ*(B,*XOR*(C,D). Ou seja, o usuário primeiro realiza a tarefa B para em seguida realizar a tarefa C ou a tarefa D. Para realizar a tarefa B – *AND*(E,F) –, o usuário tem que realizar as suas duas sub-tarefas (E e F) obrigatoriamente e em qualquer ordem. Dessa forma, o *estado B – Visão* do Statechart , possui dois sub-estados, *E – O.I.* e *F – O.I.*. Como no modelo da tarefa, as tarefas associadas a esses estados são regidas pelo operador *AND*, o sub-estado terá a estrutura apresentada na Figura (4.6). A *guarda* [escolhido()] avalia qual dos sub-estados de *B – Visão* foi escolhido para ser realizado a transição. Estando em um desses estados (*E – O.I.* ou *F – O.I.*), e ocorrendo o evento ePR, as condições do segundo estado condicional tem que ser avaliadas. Neste caso, a *guarda* [G1] (G1 = escolhidoEscolhido() e estadoNãoRealizado()) e a *guarda* [TodosSatisfeitos] tem que ser avaliadas para se verificar se a seqüência, condições e restrições relativas ao operador *AND* já foram satisfeitas. Caso todas estejam satisfeitas, o estado *B – Visão* é deixado para o estado *XOR*, caso contrário ele irá pra um dos sub-estados *B – Visão* que a guarda [G1] indicar.

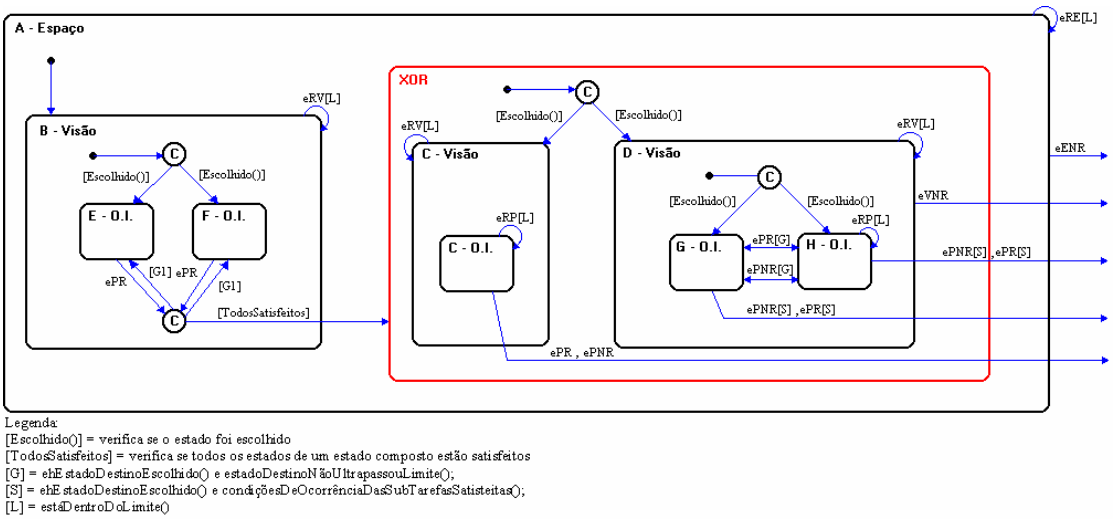


Figura (4.6): Representação Statechart do diálogo intra-EDITOR.

No estado *XOR* apenas o estado *C – Visão* ou *D – Visão* pode ser realizado, daí, as transições são representadas de acordo com a Figura (4.6). Os eventos que estão indo para fora do estado *A – Espaço*, representam transições para quaisquer outros estados do Statechart.

4.6 - Exemplo 2 – Diálogo Inter-*Editor*

Será apresentado nesta seção, um exemplo que contempla o diálogo inter-*Editores*. A Figura (4.7) representa um trecho de uma árvore de tarefa de um sistema.

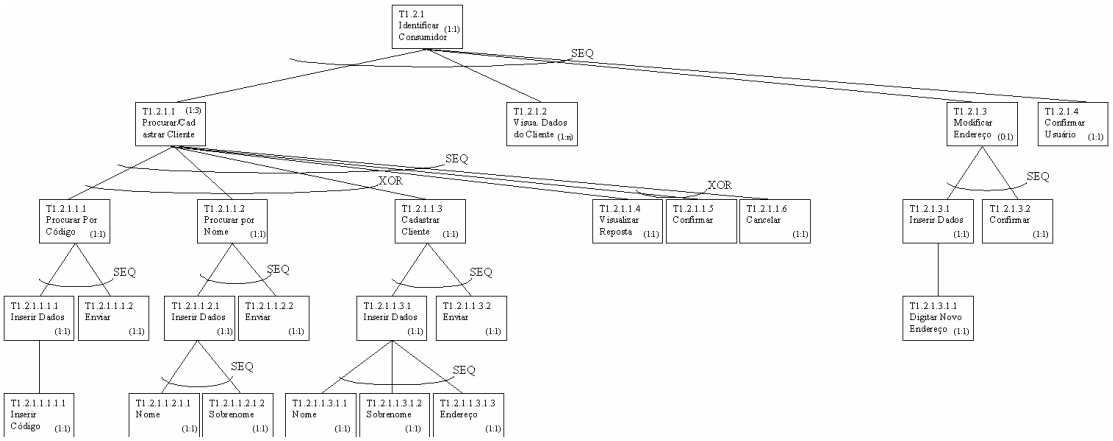


Figura (4.7): Trecho de árvore de tarefa

Quadro (4.4): Trecho de tarefa identificar consumidor

Número	Nome	Ocorrência
T1.2.1	Identificar Consumidor	(1,1)
T1.2.1.1	Procurar/Cadastrar Cliente	(1,n)
T1.2.1.1.1	Procurar por Código	(1,1)
T1.2.1.1.1.1	Inserir Dados	(1,1)
T1.2.1.1.1.1.1	Inserir Código	(1,1)
T1.2.1.1.1.2	Enviar	(1,1)
T1.2.1.1.2	Procurar por Nome	(1,1)
T1.2.1.1.2.1	Inserir Dados	(1,1)
T1.2.1.1.2.1.1	Nome	(1,1)
T1.2.1.1.2.1.2	Sobrenome	(1,1)
T1.2.1.1.2	Enviar	(1,1)
T1.2.1.1.3	Cadastrar Cliente	(1,1)
T1.2.1.1.3.1	Inserir Dados	(1,1)
T1.2.1.1.3.1.1	Nome	(1,1)
T1.2.1.1.3.1.2	Sobrenome	(1,1)
T1.2.1.1.3.1.3	Endereço	(1,1)
T1.2.1.1.3	Enviar	(1,1)
T1.2.1.1.4	Visualizar Resposta	(1,n)
T1.2.1.1.5	Confirmar	(1,1)
T1.2.1.1.6	Cancelar	(1,1)
T1.2.1.2	Visualizar Dados do Cliente	(1,n)
T1.2.1.3	Modificar Endereço	(0,1)
T1.2.1.3.1	Inserir Dados	(1,1)
T1.2.1.3.1.1	Digitar Novo Endereço	(1,1)
T1.2.1.3.2	Confirmar	(1,1)
T1.2.1.4	Confirmar Usuário	(1,1)

A partir desse trecho de árvore, identificamos as árvores EDITOR correspondentes (esta obtenção será vista no capítulo 5) ilustradas nas Figuras (4.8) a (4.13).

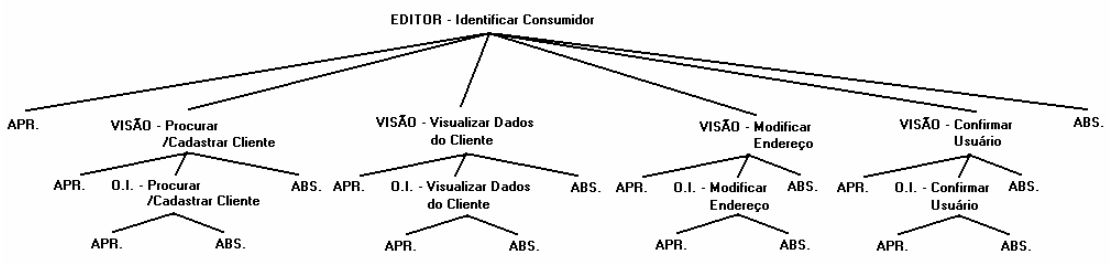


Figura (4.8): Árvore EDITOR – Identificar Consumidor

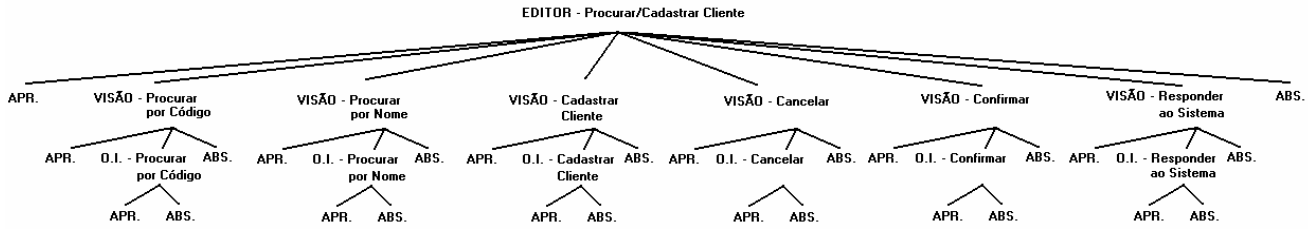


Figura (4.9): Árvore EDITOR – Procurar/Cadastrar Cliente

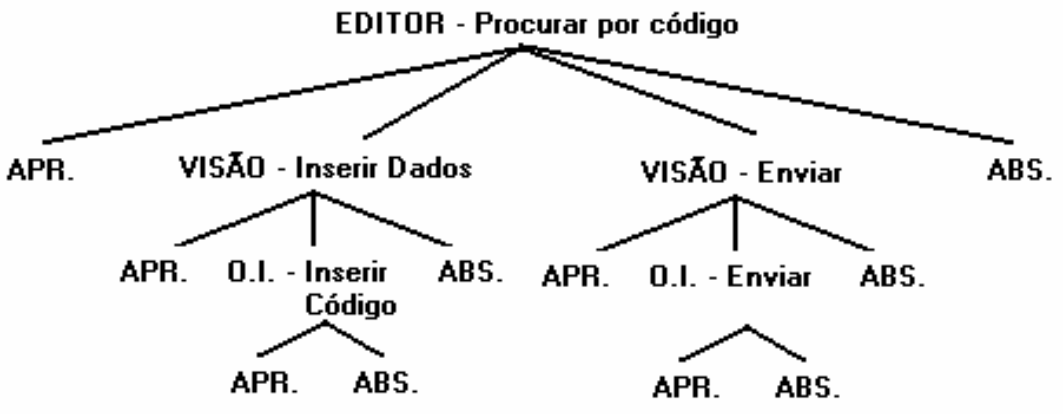


Figura (4.10): Árvore EDITOR – Procurar por Código

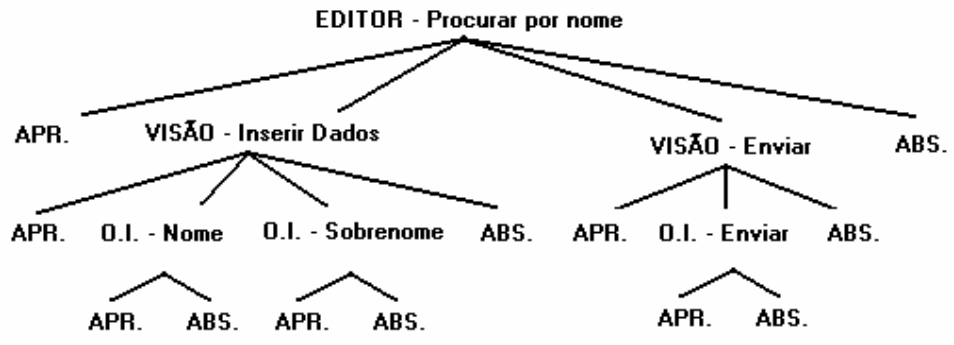


Figura (4.11): Árvore EDITOR – Procurar por Nome

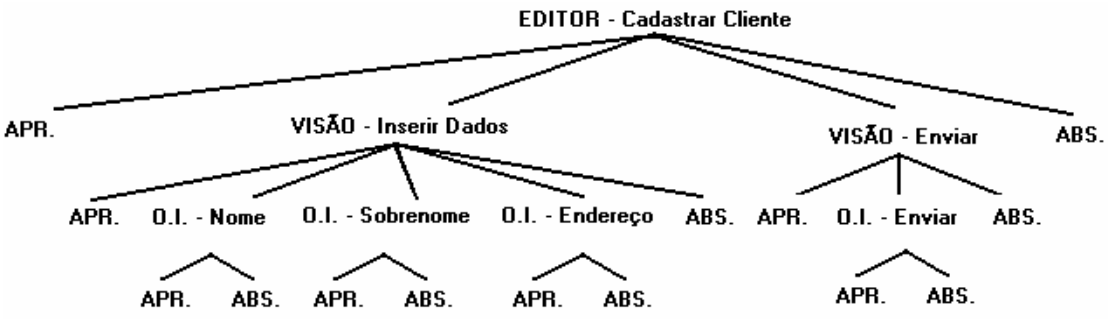


Figura (4.12): Árvore EDITOR – Cadastrar Cliente

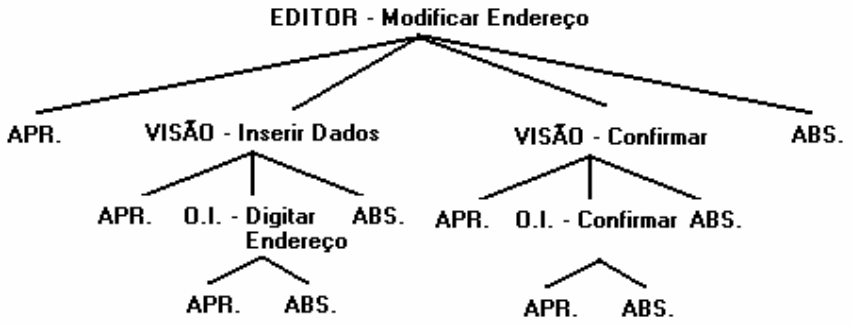


Figura (4.13): Árvore EDITOR – Modificar Endereço

De posse das Árvores EDITOR e das informações oriundas do modelo da tarefa (expressas no Quadro (4.4) e na Figura (4.7)), chegamos ao Statechart representado pela Figura (4.14). Vale salientar que os estados *E – Procurar por Nome*, *E – Cadastrar Cliente* e *E – Modificar Endereço*, estão minimizados, sendo apenas representadas as transições de entrada e de saída dos mesmos.

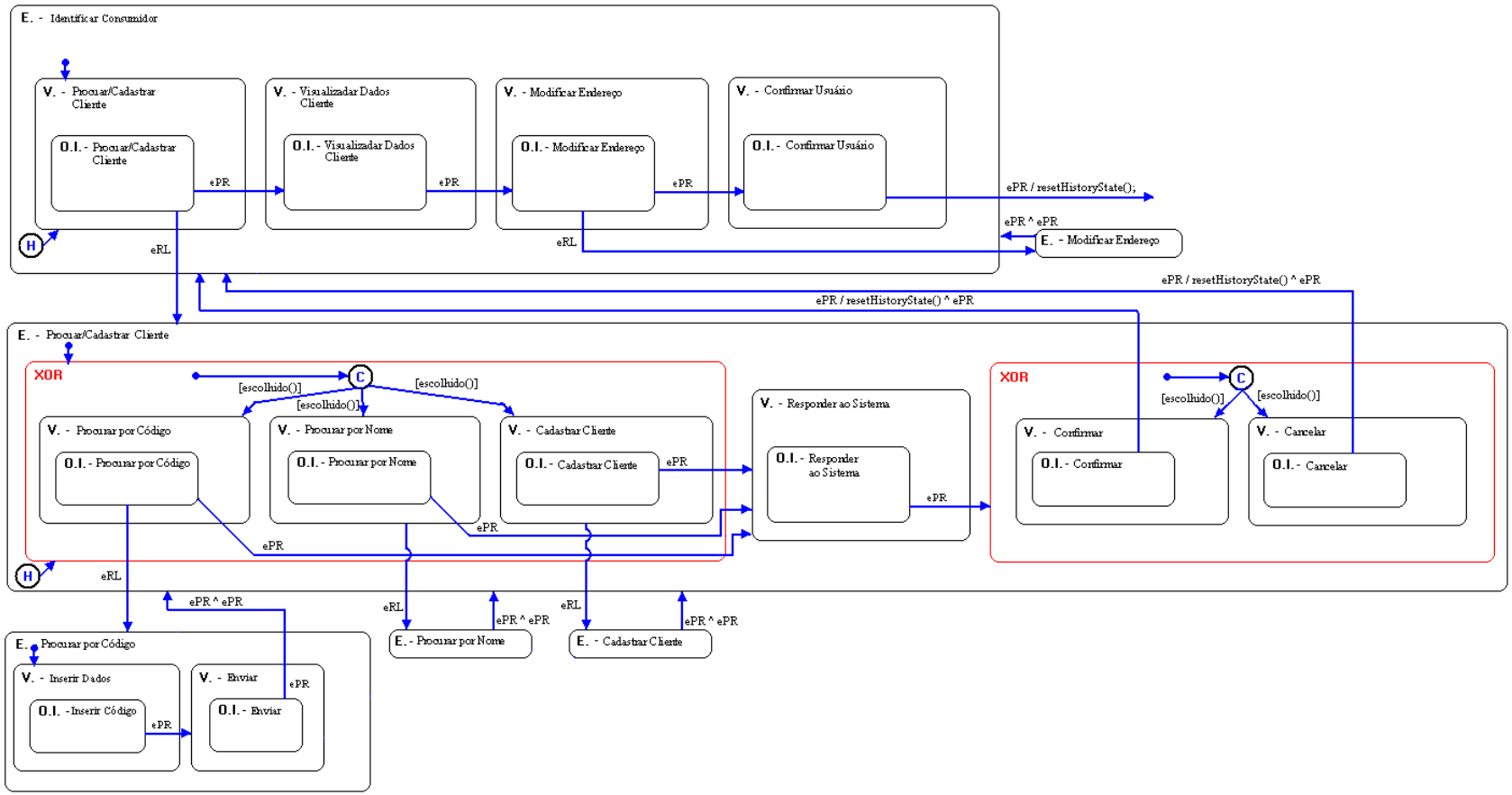


Figura (4.14): Statechart Identificar Consumidor

Os estados *O.I. – Procurar/Cadastrar Cliente* e *O.I. – Procurar por Código* respondem ao "evento Realizar Link" (eRL), que realizam transições a elementos de diferentes árvores EDITOR. Tais transições representam diálogos em estado modal, visto que uma vez ocorrida uma tal transição, o estado de origem só será novamente alcançado após o estado de destino ser totalmente realizado. Algumas observações devem ser feitas: (i) quando ocorrer uma transição em estado modal, e uma nova transição de volta ao original ocorrer, o objeto de interação terá um evento do tipo ePR ^ ePR. Esse segundo evento ePR poderá ser consumido pelo estado de destino; (ii) se o estado que estiver sendo deixado possuir um estado histórico, em uma transição modal, esse estado histórico deve ser "resetado" através do evento ePR / resetHistoryState() ^ ePR.

4.7 - Representação da Concorrência

O formalismo Statechart permite a representação da concorrência entre estados. Essa representação é feita através de estados ortogonais, ou seja, estados do tipo and-states. A Figura abaixo ilustra como representar a concorrência com Statecharts. Neste caso, a inclusão do operador *PAR* logo abaixo da raiz, quer dizer que as telas do sistema poderão ser acessadas concorrentemente.

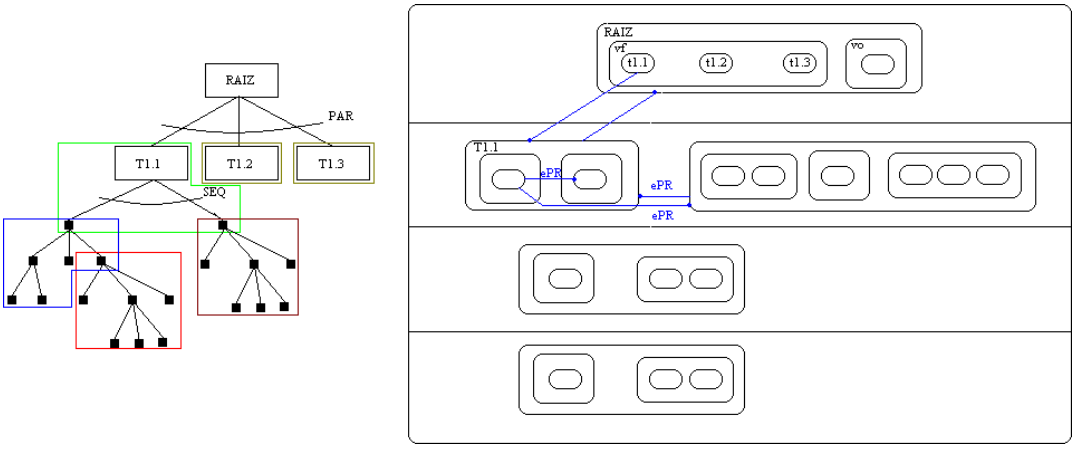


Figura (4.15): Representação Statechart de estados concorrentes

Já na Figura (4.16) abaixo, a leitura é a seguinte, todas as tarefas que tiverem abaixo do operador *PAR* serão inclusas dento de um estado ortogonal.

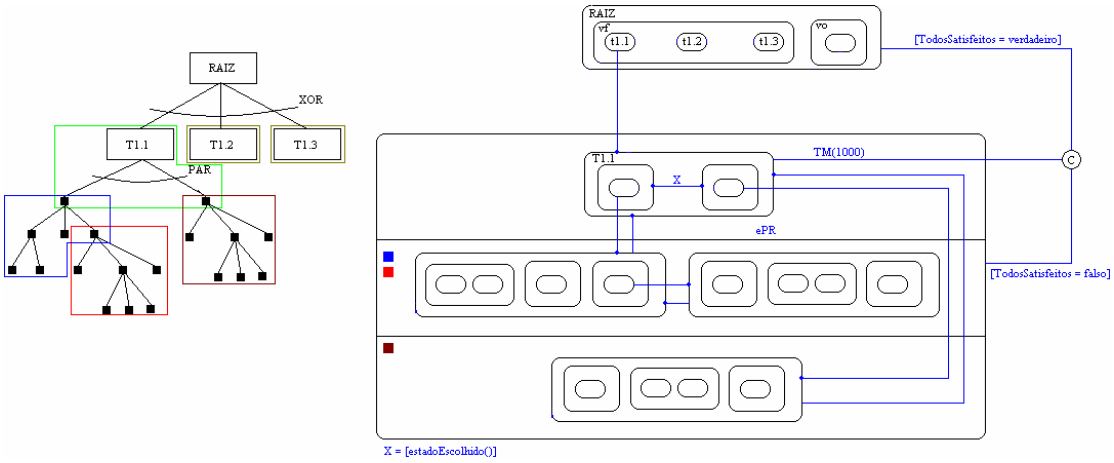


Figura (4.16): Representação Statechart de estados concorrentes

4.8 - Conclusão

Estabeleceu-se nesse capítulo, os requisitos e características desejáveis de um formalismo para a representação do componente *Diálogo* do modelo de Interação EDITOR Estendido e escolhemos, a seguir, entre os formalismos existentes na literatura, o que mais se adequava para tal fim. Entre os formalismos baseados em eventos/transições referenciados na literatura sobre interfaces homem computador, o formalismo Statechart foi o que se mostrou mais adequado a todos os requisitos e características desejadas. Além de tudo, o fato do formalismo Statechart fazer parte da definição da UML, tende a tornar mais fácil à comunicação entre todos os envolvidos no processo de concepção de interfaces.

Portanto, definido o componente de diálogo e escolhido o formalismo para sua representação, será apresentado no próximo capítulo a metodologia MEDITE^{+}, – uma variante da metodologia MEDITE que implementa a nova abordagem de aquisição do modelo da interação a partir do modelo da tarefa, permitindo uma automatização do processo de obtenção do modelo da interação.

Capítulo 5 – MEDITE^{+}

Este capítulo apresenta a metodologia MEDITE^{+}, uma variante da metodologia MEDITE. MEDITE^{+} implementa o processo de roteirização definido por Suárez (2004). Este capítulo também apresenta os algoritmos de obtenção do modelo de interação, e os mecanismos de manutenção da coerência entre os modelos de tarefa-interação.

5.1 – Introdução

A modelagem da tarefa é hoje reconhecida como um importante ponto de partida para a geração de outros modelos, visto que ela ajuda ao projeto centrado no usuário (Johnson *et al.*, 1995). Conforme apresentado no capítulo 3, os modelos presentes nas metodologias baseadas na tarefa podem ser classificados como modelos **concretos** (apresentação, diálogo, interação, etc) ou **abstratos** (tarefa, domínio, usuário, etc). Segundo Vandedonckt (Vandedonckt *et. al.*, 2003), a obtenção dos modelos **concretos** se dá a partir dos modelos **abstratos**, podendo ser realizada das seguintes maneiras: (i) derivação (modelos concretos são derivados dos abstratos a partir de conhecimento ergonômico em forma de regras), (ii) ligação/conexão (modelos concretos são concebidos e depois ligados, ou seja, relacionados, entre elementos de modelos distintos previamente definidos) ou (iii) composição (modelos já concebidos são montados, parcialmente ou totalmente, com outros modelos para reconstrução dos modelos fonte, ou para construção de novos modelos relacionamos com o mundo real).

MEDITE, em sua versão original, utiliza a estratégia de derivação (utilizando regras ergonômicas) do modelo da interação a partir do modelo da tarefa. No capítulo 2 foram apresentados os principais problemas identificados em MEDITE, entre eles a dificuldade da utilização de regras ergonômicas para a obtenção do modelo da interação a partir do modelo da tarefa. Este capítulo contém a introdução da nova abordagem de obtenção do modelo da interação na metodologia MEDITE – *processo de roteirização* (Suárez, 2004). A utilização do processo de roteirização (modelo de roteiro como "ponte" entre o modelo da tarefa e o modelo de interação) possibilita a automatização do processo de obtenção do modelo de interação (EDITOR Estendido) e a manutenção

da coerência entre os modelos. A nova variante da metodologia oriunda dessa modificação será denominada de MEDITE^{+}.

5.2 - Modelo de Roteiro

Suárez (Suárez, 2004) realizou um estudo de gestão do conhecimento no processo de concepção de interfaces homem-computador com vistas a criar uma ontologia que tentasse uniformizar os termos e conceitos presentes em diversas metodologias de desenvolvimento de interfaces e diminuir a dificuldade inerente ao processo de obtenção de uma especificação conceitual da interação. O primeiro resultado de Suárez foi a criação de três meta-modelos: tarefa, usuário e interação.

A classificação e representação dos conhecimentos obtidos nos meta-modelos supracitados, entretanto, não foram fatores suficientes para garantir uma solução para a dificuldade de integração e de utilização desses modelos no processo de obtenção de uma especificação conceitual da interação. Suárez, então, criou o meta-modelo de roteiro baseado em uma abordagem fundamentada em uma metáfora. Suárez (2004) introduziu uma metáfora cênica, onde um roteiro (*script*) seria definido como uma composição de *cenário*s, *cena*s e *tomada*s. Esse meta-modelo foi introduzido com vistas a diminuir os problemas relativos à obtenção do modelo da interação a partir do modelo de tarefas usando regras ergonômicas, e a possibilitar a automatização do processo de obtenção do modelo da interação.

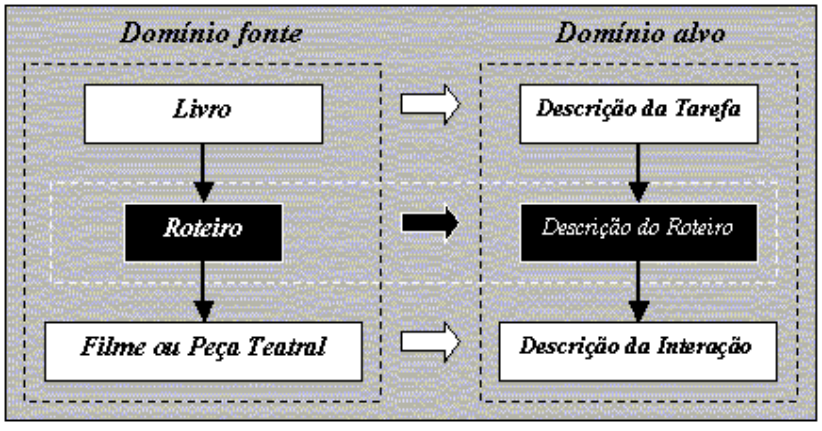


Figura (5.1): Processo de transferência de significado empregado no domínio do processo de concepção de IHC (metáfora cênica)

Portanto, para Suárez (2004), o processo de obtenção do modelo de interação a partir do modelo de tarefa (processo de roteirização) é dividido em duas etapas: (i) elementos do modelo da tarefa são mapeados em elementos do modelo de roteiro; e em seguida, (ii) elementos do modelo de roteiro são mapeados em elementos do modelo de interação. Segundo Suárez (2004), a obtenção do modelo de roteiro é realizada a partir das regras estruturais apresentadas no Quadro (5.1).

Quadro (5.1): Regras para mapear elementos do meta-modelo da tarefa em elementos do meta-modelo de roteiro

Regra	Descrição da Regra
Regra Tomada01	*Se a Tarefa é uma <Ação> (MTa) defina uma <Tomada> (MRo) associada à Tarefa*
Regra Cena01	*Se a Tarefa é uma <Tarefa> (MTa) cuja decomposição apresente pelo menos uma <Ação> (MTa), defina uma <Cena> (MRo) associada à Tarefa*
Regra Cenário01	*Se a Tarefa é uma <Tarefa> (MTa) cuja decomposição apresente pelo menos uma <Tarefa> (MTa) que esteja associada a uma <Cena> (MRo), defina um <Cenário> (MRo) associado à Tarefa*
Regra Cenário02	*Se a Tarefa é uma <Tarefa> (MTa) que está associada a uma <Cena> (MRo) e pelo menos um de seus irmãos é uma <Tarefa> (MTa) associada a um <Cenário> (MRo), associe à Tarefa um <Cenário> (MRo)*
Regra Cena02	*Se a Tarefa é uma <Ação> (MTa) que seja filha de uma <Tarefa> (MTa) que esteja associada a um <Cenário> (MRo), associe à Tarefa uma <Cena> (MRo)*

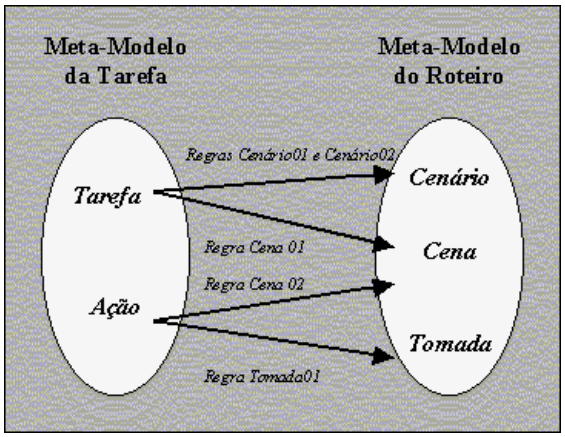


Figura (5.2): Primeiro momento associado à transformação do meta-modelo da tarefa no modelo da interação (meta-modelo da Tarefa em meta-modelo do Roteiro)

A transformação do meta-modelo de roteiro no meta-modelo de interação, é realizada de acordo com as regras do Quadro (5.2).

Quadro (5.2): Mapeamento dos elementos do meta-modelo do roteiro em elementos do meta-modelo da interação.

Mapeamento entre os conceitos	Conceito do Modelo do Roteiro	Conceito do Modelo da Interação
ObtençãoDosEspaços	Cenário	Espaço
ObtençãoDasVisões	Cena	Visão
ObtençãoDosObjetosDeInteração	Tomada	ObjetoDeInteração

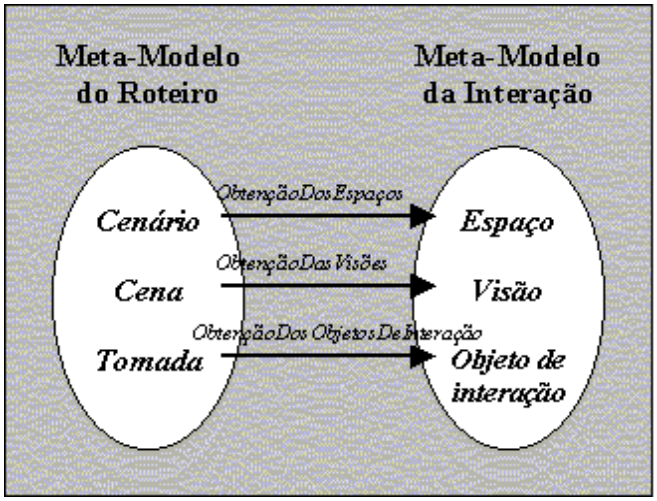


Figura (5.3): Segundo momento associado à transformação do meta-modelo de tarefa no meta-modelo de interação (meta-modelo do roteiro em meta-modelo da interação)

Dessa maneira, todo o processo é realizado conforme a Figura (5.4). Elementos do modelo de interação contêm link para o elemento do modelo de roteiro a que está associado. Por seu turno, elementos do modelo de roteiro contêm link para o elemento do modelo de tarefa a que está associado. Essa ligação possibilita a rastreabilidade entre os elementos, ou seja, há como saber em qualquer um dos elementos dos modelos (roteiro e interação), qual elemento da tarefa o originou.

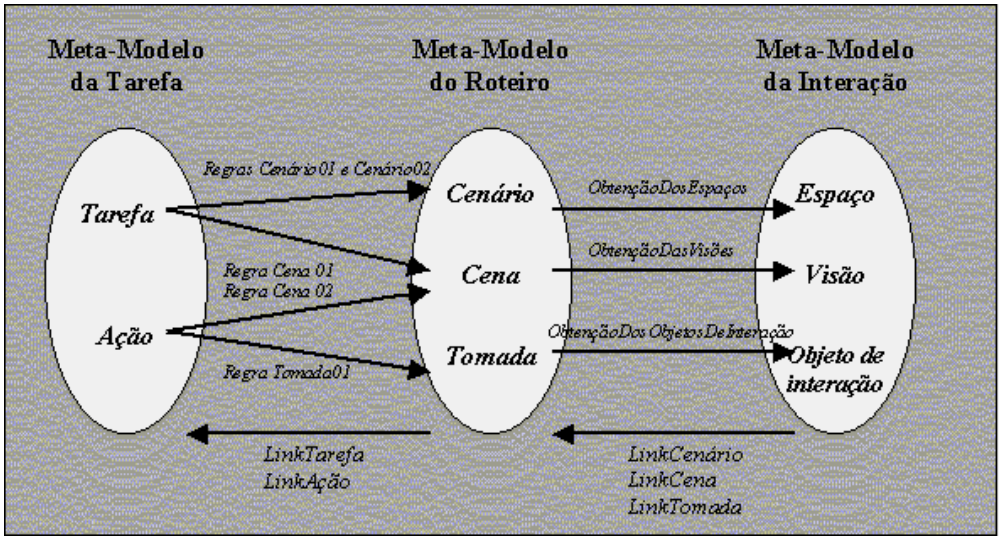


Figura (5.4): Mecanismo para a manutenção da rastreabilidade entre os elementos dos meta-modelos da tarefa, do roteiro e da interação.

A utilização da nova abordagem proporciona: (i) a possibilidade de automatização do processo de obtenção do modelo de interação, sendo sua implementação bastante simples se comprada com a utilização de regras ergonômicas e/ou de produção; e (ii) a possibilidade da construção de mecanismos que promovam a manutenção da coerência entre os modelos envolvidos (tarefa, roteiro e interação), em virtude da possibilidade de rastreabilidade entre eles.

5.3 – MEDITE^{+}

Como fora apresentado no capítulo 2, MEDITE está dividida em 5 etapas, quais sejam: (i) Análise e modelagem da tarefa; (ii) Especificação conceitual parcial da interação (construção das Árvores EDITOR); (iii) Especificação completa (definição dos atributos) ; (iv) Geração do protótipo ; e (v) Avaliação.

A introdução da nova abordagem (preconizada por Suárez (2004)) em MEDITE passa pela divisão da etapa da (ii) (Especificação conceitual parcial da interação) em duas sub-etapas, a saber: (ii.a) especificação do modelo de roteiro (Modelo de Roteiro); (ii.b) especificação conceitual parcial da interação (Modelo EDITOR Estendido). Ou seja, a substituição do processo de derivação (utilizando regras ergonômicas) pelo processo de roteirização.

Assim, a figura (5.5) apresenta a nova versão de MEDITE, que será chamada doravante neste texto de MEDITE^{+} (MEtodologia de Desenvolvimento de InTerfaces Ergonômicas).

Dessa forma, MEDITE^{+} define um processo de concepção e desenvolvimento de uma interface com as seguintes etapas:

(i) Análise e modelagem da tarefa (Modelo TAOS);

(ii) Especificação do modelo de roteiro (Modelo de Roteiro);

(iii) Especificação conceitual parcial da interação (Modelo EDITOR Estendido);

(iv) Especificação completa da interação (Modelo EDITOR Estendido);

(v) Geração do protótipo (OO); e

(vi) Avaliação.

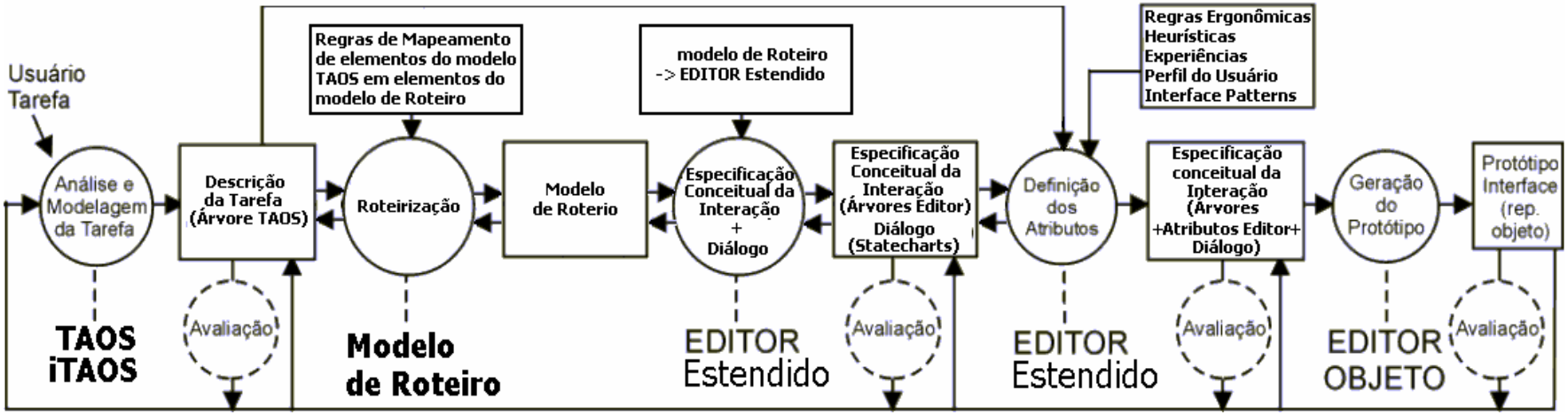


Figura (5.5): MEDITE+ com o processo de "roteirização".

5.3.1 - Análise e Modelagem da Tarefa (Modelo TAOS)

Esta etapa consiste na análise e modelagem da tarefa do usuário. O intuito da realização desta etapa é identificar precisamente a tarefa do usuário entendendo a sua lógica, ou seja, a maneira, os procedimentos e objetos que ele utiliza para executá-la. Através do usuário e do domínio da tarefa, encontram-se os requisitos necessários ao sistema que se pretende conceber. Utiliza-se nesta etapa o Modelo TAOS e a ferramenta iTAOS (Cordeiro, 2003; Medeiros, 2003).

5.3.2 - Especificação do Modelo de Roteiro (Modelo de Roteiro)

Esta etapa consiste no mapeamento automático do modelo da tarefa no modelo de roteiro. O modelo de roteiro serve como ponte entre os modelo de tarefa e de roteiro. Utilizam-se as regras apresentadas na seção 5.4.1 deste documento.

5.3.3 - Especificação Conceitual Parcial da Interação (Modelo EDITOR Estendido)

Esta etapa consiste no mapeamento automático do modelo de roteiro supracitado no modelo de interação (parcial) EDITOR. Neste ponto, o projetista obterá toda a estrutura de sua interface em forma das Árvores EDITOR, assim como o diálogo em nível abstrato através de diagramas Statecharts. Nesta etapa, ainda não há se trabalha com objetos de interação concretos. Utilizam-se as regras apresentadas na seção 5.4.2 deste documento.

5.3.4 - Especificação Completa da Interação (Modelo EDITOR Estendido)

A 4ª etapa consiste na definição dos atributos das árvores EDITOR. Esta etapa tem como entrada a árvore gerada na etapa anterior e a descrição TAOS da tarefa gerada na primeira etapa. Nesta etapa os atributos apresentação e abstração da árvore EDITOR devem ser preenchidos, através de regras ergonômicas, *interface patterns*, heurísticas. Este trabalho não aprofunda a discussão a partir deste nível, sendo apresentados apenas sugestões (arcabouços) para o desenvolvimento das demais fases.

5.3.5 - Geração do protótipo (OO);

Os protótipos são construídos através das árvores EDITOR obtidas na etapa anterior e dos diagramas Statechart para a navegação.

5.3.6 - Avaliação.

Esta atividade é distribuída em todas as etapas anteriores da metodologia e consiste na avaliação de cada produto de cada etapa.

5.3.6.1 - Avaliação da descrição da tarefa TAOS

Nesta fase, a avaliação consiste em verificar junto ao usuário se a árvore e descritores TAOS gerados correspondem à sua lógica de execução da tarefa. Cabe também ao projetista: verificar a completude e coerência da modelagem, eliminar tarefas não informatizáveis, modificar e melhorar as tarefas de acordo com a existência do novo sistema.

5.3.6.2 - Avaliação da árvore (parcial) EDITOR

Nesta etapa a avaliação consiste em verificar se as regras estruturais (seção 5.4.1) foram aplicadas corretamente. São avaliados o número de *Editores*, *Visões*, *Objetos_de_Interação*.

5.3.6.3 - Avaliação da árvore (completa) EDITOR

Nesta etapa a avaliação consiste em avaliar os atributos que foram definidos, a coerência entre as árvores EDITOR geradas e as árvores TAOS. O projetista deve verificar também a coerência entre as árvores e às regras utilizadas no processo de transformação. Nesta etapa, o modelo EDITOR já permite uma visualização (esboço) das janelas do sistema que se está projetando, que devem ser levadas ao usuário de forma que este possa participar do processo de concepção, validando dessa forma a descrição.

5.3.6.4 - Avaliação do protótipo

A avaliação do protótipo deve ser realizada junto ao usuário através de técnicas de avaliação (testes de usabilidade) ou ainda através de: inspeção por padrão, avaliação

heurística, conformidade com recomendações, exploração cognitiva, abordagem híbrida). Dependendo do tipo de problema que for encontrado o projetista poderá retornar à etapa imediatamente anterior ou mesmo à 1ª etapa.

5.4 - Obtenção do Modelo de Interação em MEDITE+

O processo de roteirização aplicado em MEDITE+ será, portanto, realizado em duas etapas: obtenção do modelo de roteiro a partir do modelo da tarefa TAOS; e obtenção do modelo de interação EDITOR Estendido a partir do modelo de roteiro.

5.4.1 - Algoritmo de obtenção do Modelo de Roteiro

O processo de obtenção do modelo de roteiro definido por Suárez é realizado através das regras apresentadas no Quadro (5.1). Entretanto, esse conjunto de regras não é suficiente para classificar árvores de tarefas com mais de cinco níveis, visto que elas levariam a uma situação em que um *espaço* seria composto por outros *espaços,* o que contraria a definição do modelo EDITOR Estendido. Para contornar esse problema e manter a consistência com o modelo, foi necessária a inclusão de duas novas regras, quais sejam: **Regra Tomada02** e **Regra Normalização**, definidas no Quadro (5.3) abaixo:

Quadro (5.3): Regras capazes de mapear elementos do meta-modelo da tarefa em elementos do meta-modelo do roteiro

Regra	Descrição da Regra
Regra Tomada01	*Se a Tarefa é uma <Ação> (MTa) defina uma <Tomada> (MRo) associada à Tarefa*
Regra Cena01	*Se a Tarefa é uma <Tarefa> (MTa) cuja decomposição apresente pelo menos uma <Ação> (MTa), defina uma <Cena> (MRo) associada à Tarefa*
Regra Cenário01	*Se a Tarefa é uma <Tarefa> (MTa) cuja decomposição apresente pelo menos uma <Tarefa> (MTa) que esteja associada a uma <Cena> (MRo), defina um <Cenário> (MRo) associado à Tarefa*
Regra Cenário02	*Se a Tarefa é uma <Tarefa> (MTa) que está associada a uma <Cena> (MRo) e pelo menos um de seus irmãos é uma <Tarefa> (MTa) associada a um <Cenário> (MRo), associe à Tarefa um <Cenário> (MRo)*
Regra Cena02	*Se a Tarefa é uma <Ação> (MTa) que seja filha de uma <Tarefa> (MTa) que esteja associada a um <Cenário> (MRo), associe à Tarefa uma <Cena> (MRo)*
Regra Tomada02	*Se a Tarefa é uma <Tarefa> (MTa) que não seja filha da Tarefa Raiz (MTa) e que esteja associa a um <Cenário> (MRo) a e uma <Cena> (MRo), associe esta Tarefa a uma Tomada (MRo) (Será no Modelo de Interação um Objeto de Link)*
Regra Normalização	*Se a Tarefa T, que não seja a raiz, possui pelo menos uma filha associada com a um <Cenário>, uma <Cena> e a um <Objeto de Interação>, associe T a um <Cenário>, uma <Cena> e a um <Objeto de Interação>.*

Dessa forma, os elementos da tarefa (TAOS) podem ser associados aos elementos do roteiro de acordo com a figura abaixo.

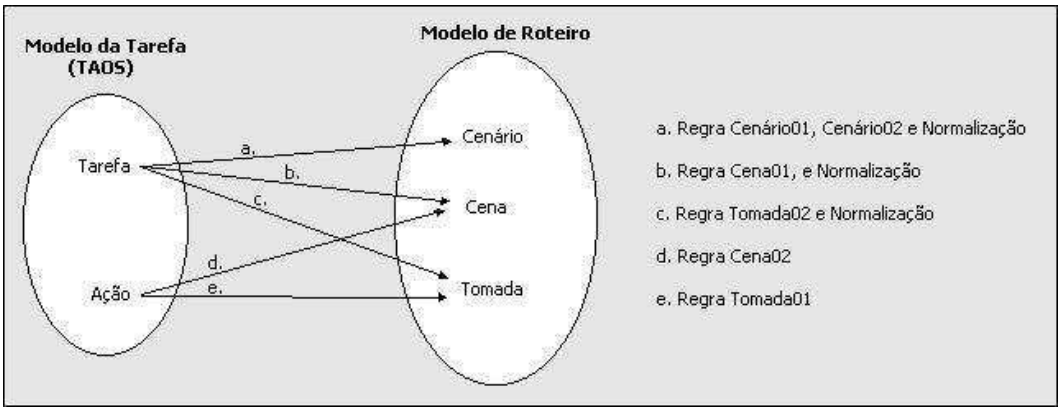


Figura (5.6): Regras para obtenção do modelo de roteiro

Assim a utilização dessas regras será efetuada segundo o algoritmo abaixo:

O algoritmo de obtenção do modelo de roteiro deve ser implementado da seguinte maneira: fazer um caminhamento em ordem na árvore, ou seja, *botton up* para cada uma das regras. Abaixo é apresentado o algoritmo (pseudo código) para obtenção dos elementos do modelo de roteiro.

```
PercorrerArvore(Regra, raiz) {

  ENQUANTO (VISITADOS TODOS NODOS DA ARVORE == FALSO) {
  SE Nodo atual tem filhos
  ENTÃO
    Visitar filho mais à esquerda não visitado
  SENÃO
    Aplicar Regra (Regra)
    Retornar ao Pai.
  }
}

PercorrerArvore(Aplica Regra Tomada01, raiz)
PercorreArvore(Aplica Regra Cena02, raiz)
PercorreArvore(Aplica Regra Cenário01, raiz)
PercorreArvore(Aplica Regra Cenário02, raiz)
PercorreArvore(Aplica Regra Cena02, raiz)
PercorrerArvore(Aplica Regra Tomada02, raiz)
PercorrerArvore(Aplica Regra Normalização, raiz)
PercorrerArvore(Aplica Regra01, raiz)
```

5.4.2 - Algoritmo de Obtenção do Modelo de Interação

A obtenção do modelo de interação EDITOR Estendido é realizada tal como fora definido por Suárez. Como visto em Suárez, a relação entre o modelo de roteiro e o modelo EDITOR estendido é uma relação bi-unívoca (um para um), onde um *Cenário* será um *Espaço (Editor)*, uma *Cena* uma *Visão* e uma *Tomada* um *Objeto de Interação*. A figura (5.7) abaixo ilustra essa relação.

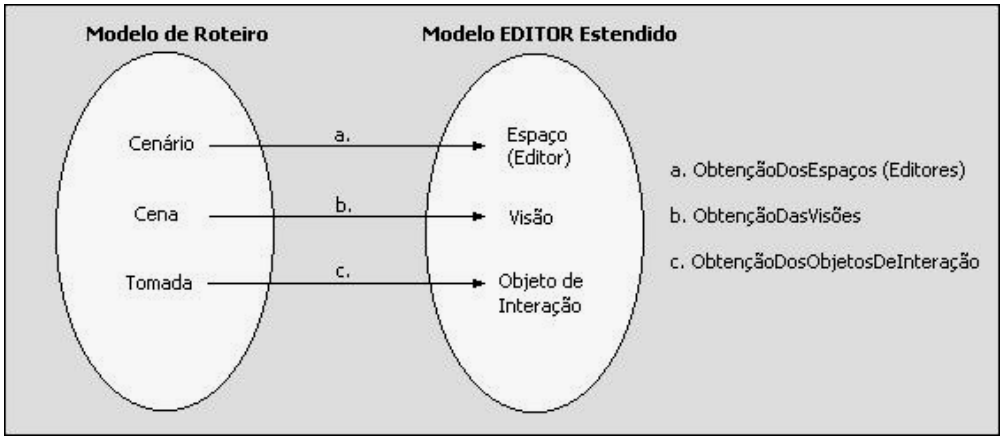


Figura (5.7): Regras para obtenção do modelo de interação

Abaixo é apresentado o algoritmo (pseudo código) para obtenção dos elementos do modelo de interação a partir dos elementos do modelo de roteiro.

Para Cada Cenário -> Espaço
Para Cada Cena -> Visão
Para Cada Tomada -> Objeto de Interação

5.5 - Manutenção da Coerência

Como apresentado na seção 5.2 acima, a introdução do modelo de roteiro possibilita a rastreabilidade entre elementos de modelos distintos. Então, com a introdução da rastreabilidade seria possível realizar alterações diretamente no modelo de interação e propagar essas mudanças até o modelo da tarefa, tornando os modelos coerentes. Ou seja, em última instância, após a avaliação da interface gerada, o projetista poderia modificar diretamente a interface (um espelho do modelo da interação), que os

mecanismos de manutenção da coerência baseados na rastreabilidade propagariam de volta suas alterações, automaticamente, até o modelo da tarefa, evitando a remodelagem manual da tarefa do usuário e todos as conseqüências (desejáveis e indesejáveis) que a modelagem manual poderia causar.

A manutenção da coerência, entretanto, só é possível se o processo de transformação do modelo da tarefa no modelo da interação atender a dois requisitos:

- Possibilidade de identificação de qual tarefa está associada a qual(is) elemento(s) de interação que se pretende manipular (inserir ou excluir).
- Possibilidade de identificação da estrutura necessária à representação de um elemento do modelo da interação no modelo da tarefa. Ou seja, é preciso saber como cada elemento do modelo da interação é representado no modelo da tarefa (que elemento da tarefa o representará (*Tarefa* ou *Ação*), qual o nível e posição que a *Tarefa* (ou *Ação*) associada ao elemento da interação deverá estar na árvore de tarefa).

A utilização do modelo de roteiro atende ao primeiro requisito, visto que, o *link* contido em cada elemento dos modelos de roteiro e interação, permite em última instância, a identificação da *Tarefa* (ou *Ação*) que o origina. Ou seja, permite a identificação da tarefa associada a cada um daqueles elementos (do roteiro e da interação).

O segundo requisito também é atendido pelo modelo de roteiro. Fazendo-se uma análise das regras de obtenção do modelo de roteiro a partir do modelo de tarefa, pode-se identificar, na própria árvore da tarefa, as tarefas (e a estrutura) que representam (que estão associadas a) cada elemento do modelo da interação. Ou seja, dá para identificar, qual tarefa estará associada a um *Espaço (Editor)*, a uma *Visão* ou a um *Objeto de interação*. Se essa identificação é possível, então existe a possibilidade da construção de algoritmos que insiram ou retirem automaticamente (da árvore da tarefa) tarefas (ou ações) que representem os elementos do modelo de interação que estejam sendo manipulados (no modelo da interação). Em outras palavras, há a possibilidade da construção de mecanismos que reflitam exatamente no modelo da tarefa as mudanças realizadas no modelo interação.

A Figura (5.8a) apresenta um trecho de tarefa que foi classificada em termos do modelo de interação (primeiramente classificada em termos do modelo de roteiro, mas por haver uma relação bi-unívoca entre roteiro e interação, o modelo de roteiro não precisa ser apresentado). Nesta figura, pode-se identificar todos os elementos do modelo

da interação (*Espaços* (*Editores*), *Visões* e *Objetos de Interação*), assim como, identificar quais elementos estão associados a quais *Espaços*. As tarefas relacionadas ao *Espaço 1*, são identificadas pela cor cinza. Já as tarefas relacionadas ao *Espaço 2* são identificadas pela cor amarela. Entretanto, há tarefas identificadas pelas cores cinza e amarela simultaneamente, isto quer dizer que essas tarefas estão associadas a elementos de diferentes *árvores editor*. Ou seja, uma tarefa marcada com a cor cinza e amarela está associada a um *Espaço* e também a uma *Visão* e a um *Objeto de Interação*. A Figura (5.8b) ilustra o esboço das telas referente ao trecho de tarefa apresentado.

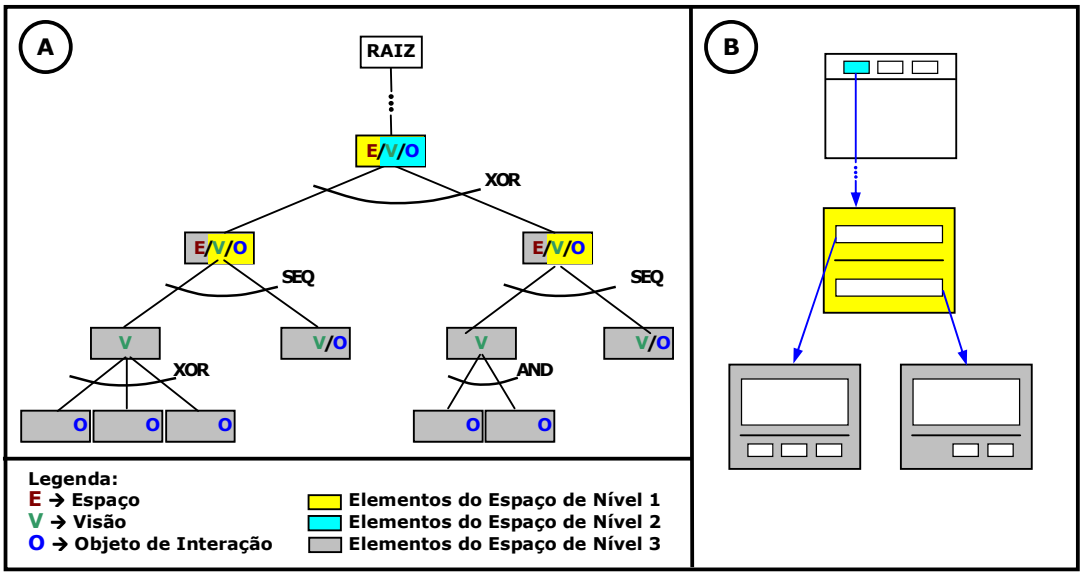


Figura (5.8): (a)Representação dos elementos do modelo de interação na árvore da tarefa TAOS; (b) Esboço das telas referentes à figura 5.8a.

Portanto, a identificação da tarefa associada ao elemento da interação e a maneira como este elemento da interação está representado na árvore da tarefa formam a base para a construção dos mecanismos (algoritmos) de manutenção da coerência entre os modelos. No entanto, para facilitar a definição desses mecanismos, utilizou-se uma classificação dos *Espaços* (*Editores*) para definir os elementos do modelo da interação que poderão ser manipulados.

5.5.1 - Classificação dos *Espaços* (Editores) no Modelo EDITOR Estendido

Um *Espaço* pode ser utilizado para diferentes tipos de interações. Por exemplo, um *Espaço* poderia servir apenas para direcionar os usuários para outros *Espaços* onde as

interações propriamente ditas seriam realizadas, ou então, ser usado para a realização das próprias interações. Por outro lado, o *Espaço* inicial da interação, além de servir como direcionador para outros *Espaços*, ele concentra os pontos de acesso às principais funcionalidades do sistema. Assim, pode-se classificar os *Espaços* de acordo com estas operações para facilitar a construção dos algoritmos de manutenção de coerência, uma vez que, cada um deles apresenta características distintas. Doravante, os *Espaço*s serão classificados como: *Espaço Inicial*, *Espaço de Direcionamento* e *Espaço de Interação*.

Espaço Inicial: Representa a tela inicia da aplicação. Contém duas visões:

1. *Visão Funcionalidade* – Contém *objetos de interação* que representarão os *links* a *Espaços de Interação* ou *de Direcionamento*.
2. *Visão Orientação* – Esta *Visão* contém um *Objeto de Interação* para exposição de algum conteúdo (inicial) ao usuário da aplicação, como, por exemplo, uma mensagem (pode ser um texto, uma figura, texto com figura, etc). A *Visão Orientação* será sobreposta pelos outros *Espaço*s da aplicação, quando eles forem acionados através da *Visão Funcionalidade*, criando assim o conceito de *Área de Trabalho*.

A figura (5.9) ilustra um *Espaço* do tipo *Espaço Inicial*.

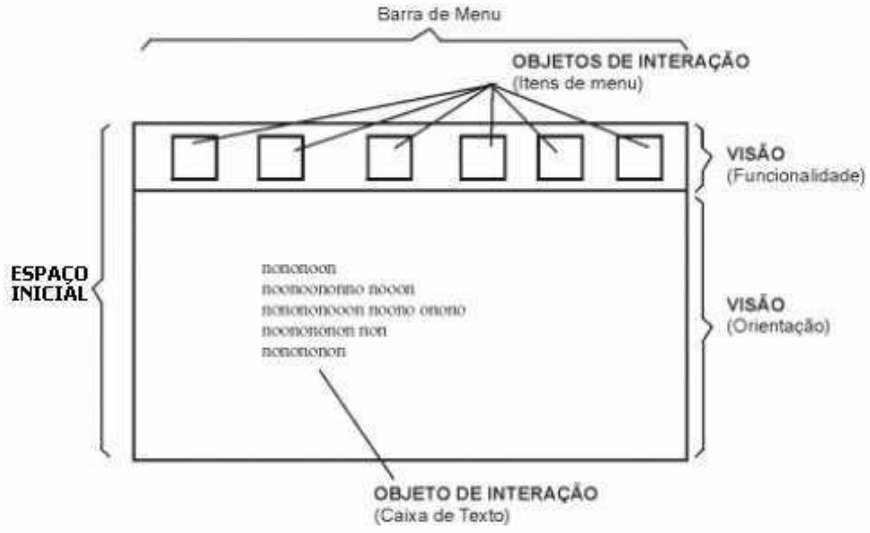


Figura (5.9): Representação de um *Espaço* do "Tipo" *Espaço* Inicial.

Espaço de Direcionamento: O *Espaço de Direcionamento* é um *Espaço* cuja tarefa a ele associada é pai de tarefas que estejam associadas a outros *Espaços* (todas as suas sub-tarefas estão associadas a *Espaços*). Ou seja, é um *Espaço* "pai" de outro(s) *Espaço*(s).

Como o nome sugere, ele serve apenas para prover o direcionamento (a navegação) do de um *Espaço* até outros *Espaços de Interação* (serão definidos adiante). Dependendo da profundidade da árvore da tarefa, poderá haver um *Espaço* de *Direcionamento* que seja "pai" de outro *Espaço de Direcionamento*. Pode-se afirmar, também, que os *Objetos de Interação* desse tipo de *Espaço* não estão associados a tarefas elementares. A figura (5.10) representa tarefas que estão associadas a um *Espaço de Direcionamento*.

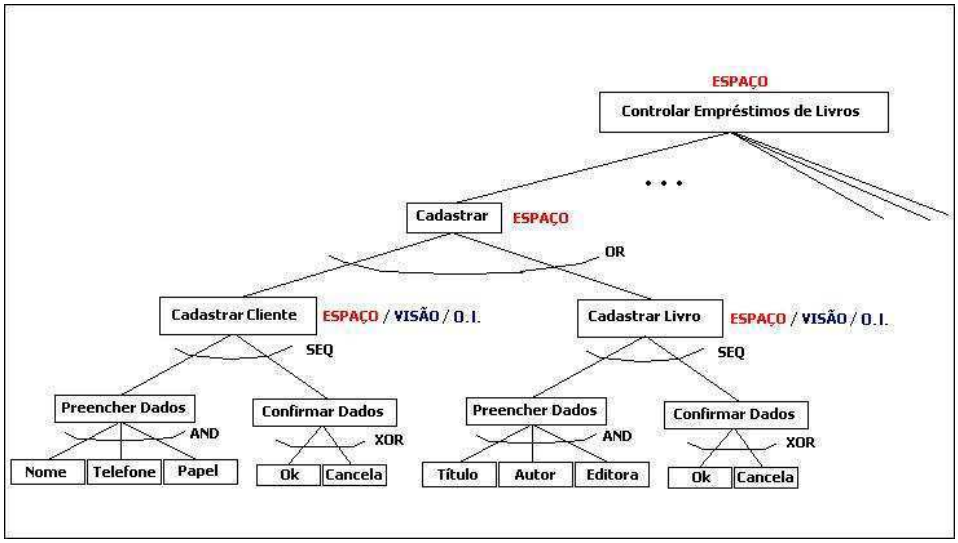


Figura (5.10): Identificando um *Espaço* de Direcionamento.

O *Espaço de Direcionamento* Cadastrar será representado de acordo com a figura (5.11).

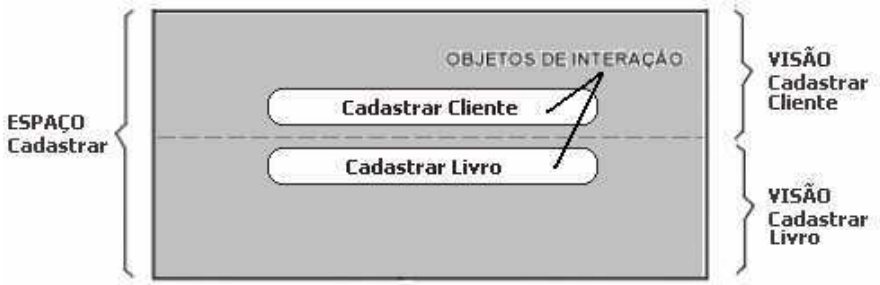


Figura (5.11): Representação do *Espaço de Direcionamento* Cadastrar.

A figura (5.12) ilustra o conceito de área de trabalho, o *Espaço* de *Direcionamento* Cadastrar sobrepõe a *Visão Orientação* do *Espaço Inicial*.

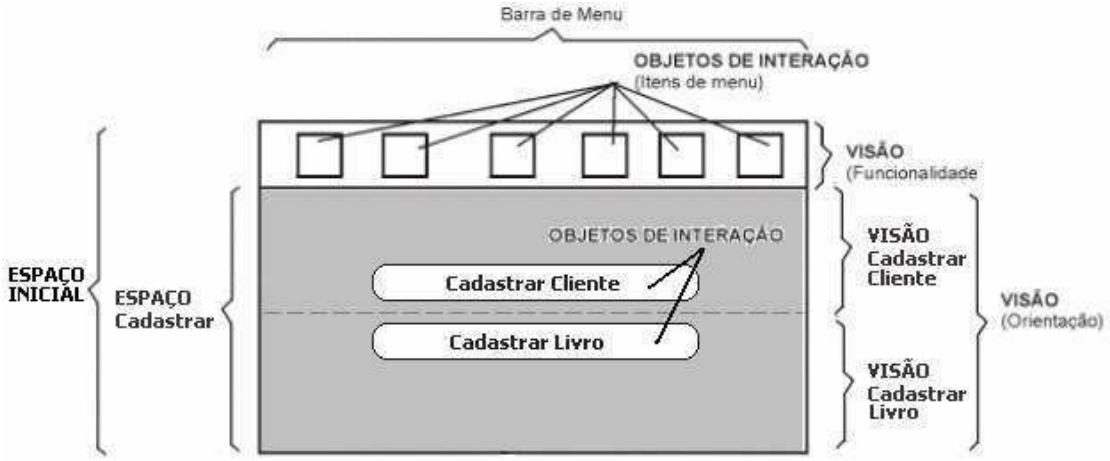


Figura (5.12): *Espaço de Direcionamento* Cadastrar sobrepondo a *Visão Orientação* do *Espaço Inicial*.

Espaço de Interação: Pelo menos um de seus elementos (*objeto de interação* e/ou *visão*) é associado a uma tarefa elementar da aplicação. É formado por *Visões* (associadas a *tarefas elementares* ou *não elementares*) e *Objetos de Interação* (associadas a *tarefas elementares* ou *não elementares*). Um típico exemplo desse tipo de *Espaço* é apresentado na figura (5.13) – esboço de tela que de um formulário.

Como apresentado na figura (5.13) o *Espaço de Interação* Cadastrar Aluno será representado por duas *Visões*, *Visão* Preencher dados e *Visão* Confirmar dados. A figura 5 ilustra o *Espaço de Interação* Cadastrar Aluno.

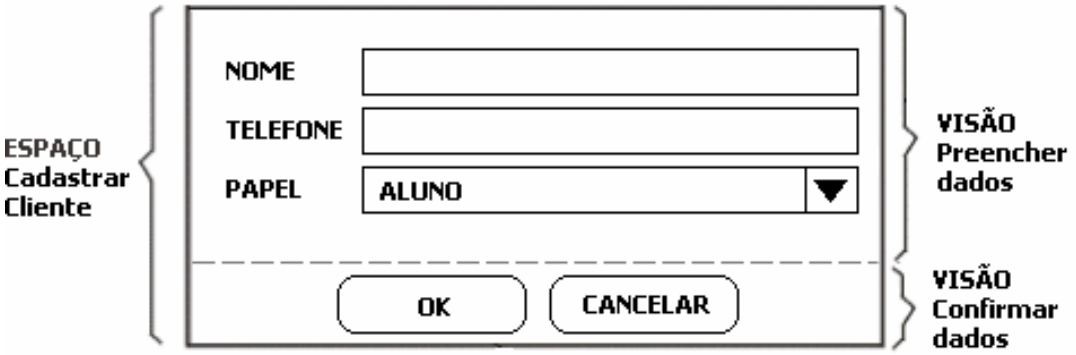


Figura (5.13): *Espaço de Interação* Cadastrar Aluno.

5.5.2 - Definição das Operações de Manipulação de Elementos do Modelo da Interação

As funcionalidades básicas de manipulação de elementos do modelo da interação e que serão objeto de propagação de volta ao modelo da tarefa são as seguintes:

1. Inclusão
   - 1.1. Incluir *Objeto de interação* (*Espaço de Interação*)
   - 1.2. Incluir *Visão (Espaço de Interação)*
   - 1.3. Incluir *Espaço de Interação*
     - 1.3.1. filho do *Espaço Inicial*
     - 1.3.2. filho de *Espaço de Interação*
     - 1.3.3. filho de *Espaço de Direcionamento*
   - 1.4. Incluir *Espaço de Direcionamento* com rearranjo (de outros *Espaços*)
     - 1.4.1. filho do *Espaço Inicial*
     - 1.4.2. filho do *Espaço de Direcionamento*
2. Exclusão
   - 2.1. Excluir *Objeto de interação* (se houver link, excluir também o *espaço* associado).
   - 2.2. Excluir *Visão*
   - 2.3. Excluir *Espaço de Interação*
     - 2.3.1. Filho do *Espaço Inicial*
     - 2.3.2. Filho de *Espaço de Direcionamento*
     - 2.3.3. Filho de *Espaço de Interação*
   - 2.4. Excluir *Espaço de Direcionamento* (sem excluir seus filhos)
     - 2.4.1. Excluir *Espaço de Direcionamento* Re-arranjando filhos no *Espaço Inicial*
     - 2.4.2. Excluir *Espaço de Direcionamento* Re-arranjando filhos no *Espaço de Direcionamento*

Com estas funcionalidades básicas, o projetista pode realizar as seguintes modificações no modelo da interação e que serão refletidas de acordo no modelo da tarefa:

- re-agrupar duas ou mais funcionalidades em um mesmo *Espaço de Direcionamento* usando a funcionalidade Inclusão de *Espaço* de Direcionamento com Rearranjo;
- inserir uma nova funcionalidade diretamente do *Espaço Inicial,* em um *Espaço de Direcionamento* ou em um *Espaço de Interação;*
- incluir *Visões* e *Objetos de Interação* (que realizem um procedimento, e não façam link a outros espaços) em um *Espaço de Interação.*
- excluir uma funcionalidade que seja acessada a partir de um *Espaço Inicial,* a um *Espaço de Direcionamento* ou em um *Espaço de Interação;*
- excluir um espaço de direcionamento, ou seja, desagrupar funcionalidades concentradas em um espaço de direcionamento

5.5.3 - Mecanismos de Manutenção de Coerência entre os Modelos

A construção dos mecanismos de manutenção da coerência, como anteriormente mencionado, irá permitir que o projetista de interfaces, de posse do modelo de interação EDITOR Estendido, possa fazer manipulações diretamente nesse modelo e essas alterações sejam propagadas automaticamente no modelo da Tarefa. Entretanto, em alguns casos, como na inclusão de novos elementos na interação, alguns atributos da tarefa, como pré e pós-situações, terão que ser posteriormente preenchidos no modelo da tarefa pelo projetista, uma vez que essas informações não estão disponíveis quando da manipulação do modelo da interação. De toda forma, esses mecanismos representam um enorme ganho de produtividade para o projetista, visto que o mesmo não precisará re-projetar e refazer manualmente a análise da tarefa toda vez que alterar o modelo de interação.

Será apresentado nesta seção um mecanismo (algoritmo) de manutenção da coerência, de inclusão de elementos no modelo da interação (mais precisamente um *Objeto de Interação* que realiza um procedimento em um *Espaço de Interação*). Os demais (12) algoritmos estarão disponíveis para verificação no **anexo B**.

Portanto, se o projetista desejar incluir um *Objeto de Interação* para realizar um procedimento (só aplicável em um *Espaço de Interação*), como por exemplo, calcular o total de uma venda, o mecanismo de manutenção da coerência a ser utilizado é o mecanismo **Incluir *Objeto de interação*** apresentado abaixo.

Incluir - *Objeto de Interação* {Objeto de interação realiza um procedimento}

MAPEAMENTO INTERAÇÃO -> ROTEIRO

1. Identificar o *Espaço E (de Interação)* onde o objeto será inserido *(Mint.).*
   1.1. Identificar o *Cenário C(MRo.)* associado ao *E(Mint.).*
2. Identificar a *Visão Vx* de *E* onde o objeto será inserido*(Mint.).*
   2.1. Identificar a *Cena Cnx (MRo.)* associada à *Vx (Mint.).*
3. Criar um *Objeto de Interação Oix*
4. SE a *Visão Vx(Mint.)* tiver um *Objeto de Interação default* (vazio) *Oid (Mint.)*
   ENTÃO Substituir Oid por *Oix* na *Visão Vx* de *E (Mint.).*
   SENÃO Inserir o *Objeto de Interação Oix* na *Visão Vx* de *E (Mint.).*
5. Criar uma *Tomada Tox (MRo.).*
   5.1. Inserir *Tox(MRo.)* como filha de *Cnx(MRo.).*
6. Associar o *Objeto de Interação Oix (Mint.)* à *Tomada Tox (MRo.).*
7. Definir a ocorrência e o Método de Realização de *Oix (Mint.)*

MAPEAMENTO ROTEIRO-> TAREFA

8. SE A *Cena CNx* (MRo.) tiver apenas uma *Tomada Tox* (MRo.) e *Tox* estiver associada a uma *Tarefa* (antes de inserir a nova tomada)
   ENTÃO
   8.1. Lançar exceção informando que a *visão Vx* associada à *Cena Cnx* não pode receber mais um objeto de interação pois já abriga um objeto de interação que faz link a outro Espaço*.

9. SE A Cena CNx (MRo.) tiver apenas uma Tomada Tox (MRo.) (antes de inserir a nova tomada)
   ENTÃO
   9.1. Identificar a *Ação Ax (MTa.)* associada à *Cena Cnx (MRo.)*
   9.2. Identificar a *Tarefa TM (MTa)* mãe da *Ação Ax (MTa.)*
   9.3. Inserir em *TM* uma *Tarefa Tx (MTa.)*
   9.4. Associar *Tx (MTa.)* à *Cnx (MRo.)* e Desassociar Ax *(MTa.)* de Cnx *(MRo.)*
   9.5. Retirar *Ax (MTa.)* de TM *(MTa.)* e inserir-la em TX *(MTa.)*
   9.6. Criar uma *Ação Af (MTa.)* como filha de *Tx (MTa.)*
   9.7. Associar *Tox (MRo.)* à *Af (MTa).*
   9.8. Modificar o atributo *Método* da *Tarefa Tx (MTa.)* inserindo *Ax e Af (MTa.).*.
   9.9. Extrair o diálogo *de Tx (MTa.)* através do atributo *Método* e *Ocorrência.*
10. SE a *Cena Cnx (MRo.)* tiver mais de uma *Tomada Tox (MRo.)* (antes de inserir a nova tomada)
    ENTÃO
    10.1. Identificar a *Tarefa Tx (MTa.)* associada à *Cnx (MRo.)*
    10.2. Criar e Associar uma *Ação Ax (MTa.)* para a *Tomada Tox (MRo).*
       10.2.1. *Criar e Inserir Pré-Situação de Ax (MTa.).*
       10.2.2. *Criar e Inserir Pós-Situação de Ax (MTa.).*

10.3. Adicionar a *Ação Ax (MTa.)* como filha da *Tarefa Tx (MTa.).*

10.3.1. Modificar o atributo *Método* da *Tarefa Tx (MTa.)* inserindo *Ax (MTa.).*.

10.3.2. Extrair o diálogo *de Tx (MTa.)* através do atributo *Método* e *Ocorrência.*

*Há uma exeção neste caso. Se a visão possuir um objeto de interação que faça link a outro espaço, então ela não pode conter mais nenhum objeto de interação.

Exemplo: O objetivo e incluir um *Objeto de Interação* em uma *Visão* de um *Espaço de Interação*. Portanto, o experimento realizado ilustrando este exemplo foi o seguinte: Acrescentar um *Objeto de Interação* na *Visão inserir dados* no *Espaço de Interação cadastrar cliente*, para registrar o telefone de um novo consumidor. No modelo da interação, a *Visão inserir dados* receberia mais um *Objeto de interação*.

A figura (5.14) ilustra o trecho da árvore da tarefa antes da inclusão do objeto de interação.

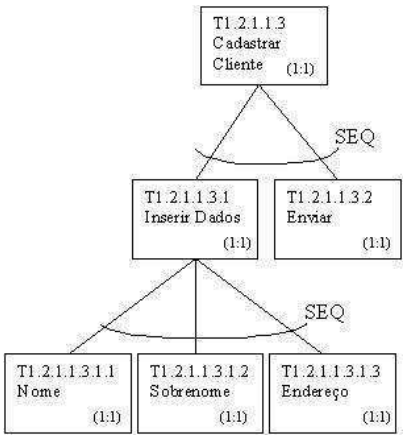


Figura (5.14): Cadastrar Cliente antes da inclusão

A figura (5.15) ilustra a árvore EDITOR (*Espaço de Interação cadastrar cliente*) correspondente ao trecho de tarefa acima apresentado.

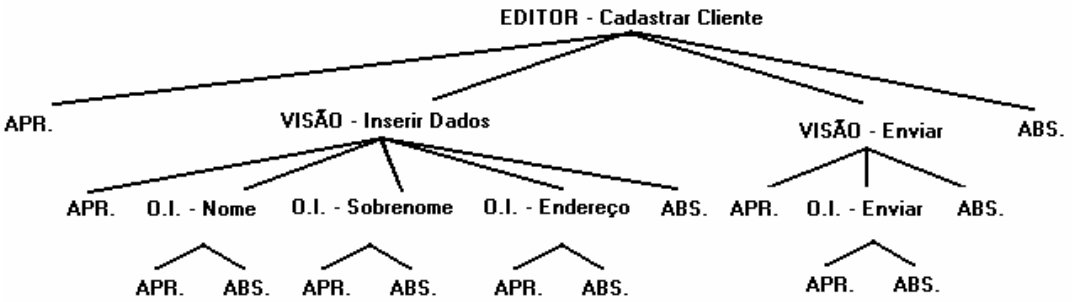


Figura (5.15): Árvore EDITOR – Cadastrar Cliente

Após a aplicação do algoritmo de manutenção da coerência entre os modelos, as estruturas dos modelos envolvidos são alteradas. A nova árvore EDITOR - Cadastrar Cliente, oriunda dessa inclusão é ilustrada na figura (5.16).

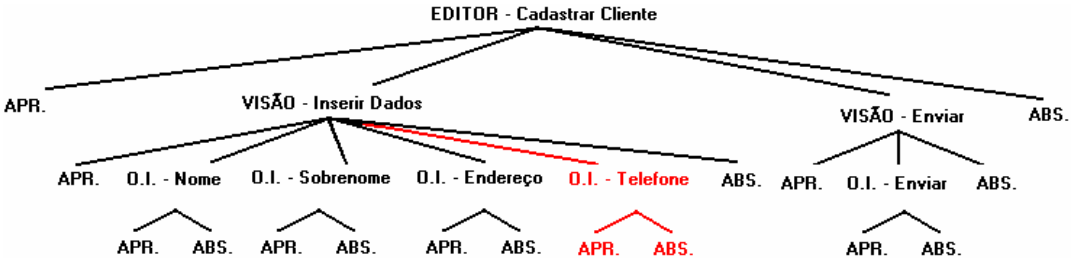


Figura (5.16): Árvore EDITOR Cadastrar Cliente após a inclusão de um *Objeto de Interação*.

Por fim, figura (5.17) apresenta o trecho da árvore da tarefa após da inclusão do objeto de interação.

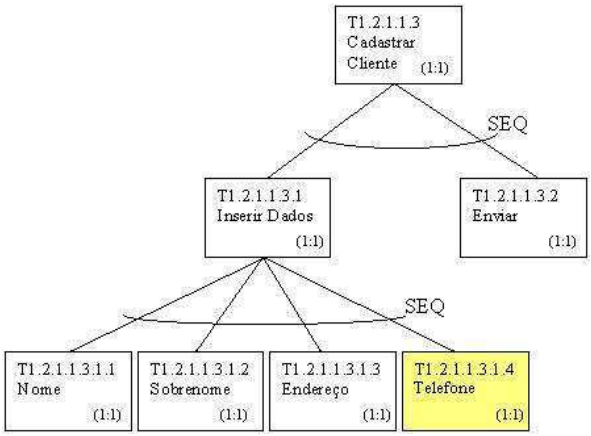


Figura (5.17): Cadastrar Cliente após a inclusão

O objeto de interação foi incluso no modelo os efeitos dessa alteração foram propagadas até o modelo da tarefa.

Como dito anteriormente, os demais algoritmos de manutenção de coerência entre os modelos podem ser consultados no **anexo B.**

5.6 - Conclusão

Foi demonstrado nesse capítulo que a utilização do processo de "roteirização" (modelo de roteiro como "ponte" entre o modelo da tarefa e o modelo de interação) possibilita a automatização do processo de obtenção do modelo de interação (EDITOR

Estendido) e possibilita a implementação de mecanismos de manutenção da coerência entre os modelos de interação e da tarefa. Dessa forma, o projetista poderá fazer modificações diretamente no modelo de interação e estas serão sempre propagadas corretamente até o modelo da tarefa. Em alguns casos, entretanto, o projetista necessitará preencher alguns atributos da tarefa manipulada visto alguns atributos não podem ser derivados no momento da manipulação de um elemento no modelo da interação, como, por exemplo, pré e pós-situações. Contudo, esses mecanismos representam um ganho enorme de produtividade para o projetista, visto que o mesmo não precisará re-projetar e refazer manualmente a modelagem da tarefa toda vez que fizer alterações no modelo de interação. Nenhuma das metodologias citadas nos capítulos 1 e 2 deste trabalho apresentam mecanismos de manutenção da coerência entre modelos. Em trabalho recente Clerckx, Luyten e Coninx (Clerckx *et al.*, 2004) apresentam uma nova metodologia (DynaMO-AID) que incorpora mecanismos de manutenção da coerência entre os modelos da tarefa e da interação.

O próximo capítulo tratará do projeto e implementação dos algoritmos de obtenção do modelo de roteiro e da interação, assim como dos mecanismos de manutenção de coerência entre os modelos envolvidos.

Capítulo 6 – Projeto e Implementação

Neste capítulo serão apresentados os elementos referentes ao projeto e implementação: (i) dos algoritmos de obtenção do modelo de roteiro e da interação, (ii) do componente de diálogo da aplicação (apresentando nos capítulo 3 e 4) e (iii) dos mecanismos de manutenção de coerência entre os modelos (tarefa, roteiro e interação) da metodologia MEDITE$^+$ (capítulo 5).

6.1 - Introdução

Conforme apresentado no capítulo 2, um dos problemas relativos à utilização de muitas metodologias de concepção de interfaces é a ausência de suporte ferramental para modelagem de seus artefatos. A ausência deste suporte acarreta a necessidade de realização manual dos artefatos, dificultando o trabalho dos projetistas. As modificações na metodologia MEDITE propostas neste trabalho (capítulo 5) serão acompanhadas por uma solução de implementação, que, por seu turno, servirão como base para a construção de uma ferramenta completa (a ser implementada em trabalhos futuros) para auxiliar as etapas 2 e 3 de MEDITE$^+$.

6.2 - Projeto

Como foi apresentado ao longo deste trabalho, MEDITE$^+$ utiliza 3 modelos: modelo da tarefa (TAOS), modelo de roteiro (Roteiro) e modelo da interação (EDITOR Estendido). Todos os modelos de MEDITE$^+$ contêm uma ligação para o modelo adjacente, o que possibilita a obtenção automática do componente de diálogo e a construção dos mecanismos de manutenção da coerência. A partir dos elementos do modelo da tarefa, sabe-se quais elementos do modelo de roteiro serão gerados, assim como, a partir dos elementos do modelo de roteiro, sabe-se quais elementos do modelo de interação serão gerados. As Figuras (6.1) e (6.2) ilustram os modelos e os links no sentido tarefa-interação e interação-tarefa, respectivamente.

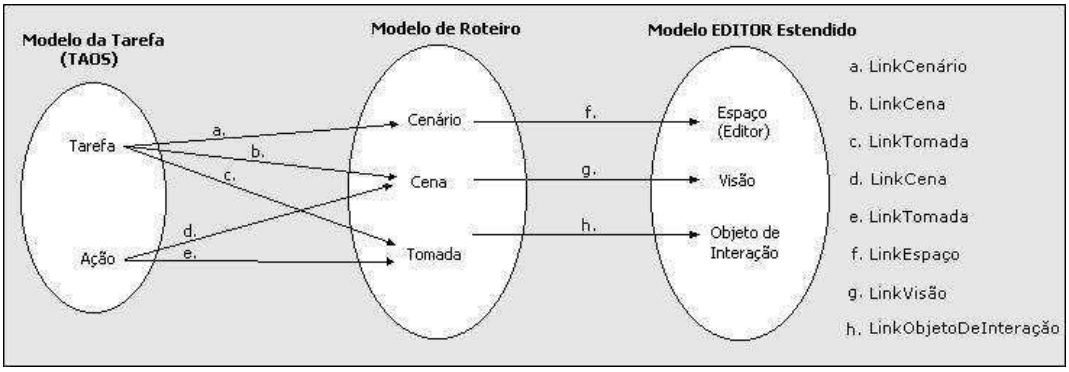


Figura (6.1): Modelos de MEDITE^{+} e os *links* entre Tarefa-Roteiro e Roteiro-Interação

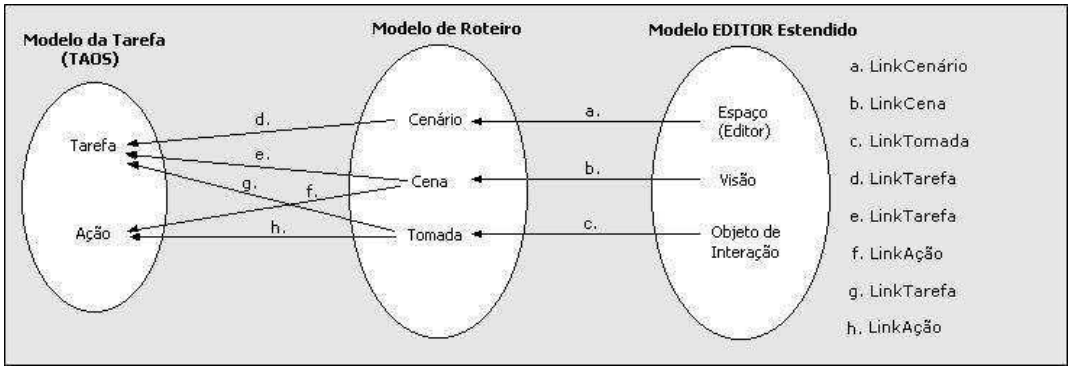


Figura (6.2): Modelos de MEDITE^{+} e os *links* entre Interação-Roteiro e Roteiro-Tarefa

Para a aplicação dos algoritmos propostos no capítulo anterior, assim como para a obtenção do componente de diálogo da aplicação, foi necessária a definição das classes que representam os modelos e os links entre eles.

O modelo da interação é composto de *Espaços* (*Editores*), *Visões* e *Objetos de Interação*. Um *Espaço* contém n *Visões*, uma *Visão* contém m *Objetos de Interação*. Todo elemento do modelo de interação tem uma ligação a um estado no statechart que representa o componente de diálogo. Embora o modelo EDITOR Estendido possua um maior número de elementos que certamente refletiriam na criação de mais classes, para as necessidades deste trabalho (especificação conceitual parcial da interação), apenas as classes identificadas acima são suficientes. A Figura (6.3) ilustra, entre outras, as classes que implementam o modelo da interação considerado.

O modelo de roteiro é análogo ao modelo da interação e é composto de *Cenários*, *Cenas* e *Tomadas* (um *Cenário* contém n *Cenas* e uma *Cena* contém m *Tomadas*). A Figura (6.3) apresenta, entre outras, as classes que implementam o modelo de roteiro.

A metodologia MEDITE^{+} utiliza o formalismo TAOS para representação e modelagem da tarefa. O formalismo TAOS dispõe de uma ferramenta (iTAOS) para modelagem da tarefa. A ferramenta iTAOS é de código aberto (seu código é disponível à comunidade de desenvolvedores de software) e foi desenvolvida em dois módulos: módulo TAME (Cordeiro, 2003), parte funcional da ferramenta; e o módulo TAOS-Graph (Medeiros, 2003), interface gráfica que permite acesso as funcionalidades do módulo TAME. Como o módulo TAME estava pronto e disponível, ele foi escolhido para a manipulação dos elementos do modelo da tarefa havendo então um re-aproveitamento de código.

Entretanto, para a utilização direta do módulo TAME, seria necessária a extensão das classes de representação das *tarefas* e das *ações* para que elas contivessem *links* aos elementos do modelo de roteiro. Tal modificação não pode ser promovida, pois alterações nas classes de representação das *tarefas* e das *ações* causariam um impacto na persistência do modelo de tarefa, necessitando promover alterações na entrada padrão (arquivo XML) do módulo TAME. Por isso, um modelo simplificado da tarefa (Figura (6.3)), chamado *ModeloTarefaVirtual* foi criado para superar esse problema. Depois de aplicado o algoritmo de obtenção do roteiro, uma classe do *ModeloTarefaVirtual* é criada, recebendo como parâmetro o modelo de tarefa real (módulo TAME), criando um modelo de tarefa virtual (de ligação ou temporário) com *links* para o módulo TAME e para o modelo de roteiro. Portanto, operações que manipulem elementos do modelo da tarefa realizam modificações no modelo de tarefa virtual e logo em seguida propagam estas modificações no módulo TAME. Esta solução permitiu o não comprometimento da persistência do modelo da tarefa, prevenindo alterações no DTD (Document Type Definition) do modelo da tarefa gerado pelo módulo TAME.

A Figura (6.3) abaixo, mostra a estrutura básica das classes que representam os modelos envolvidos em MEDITE^{+}.

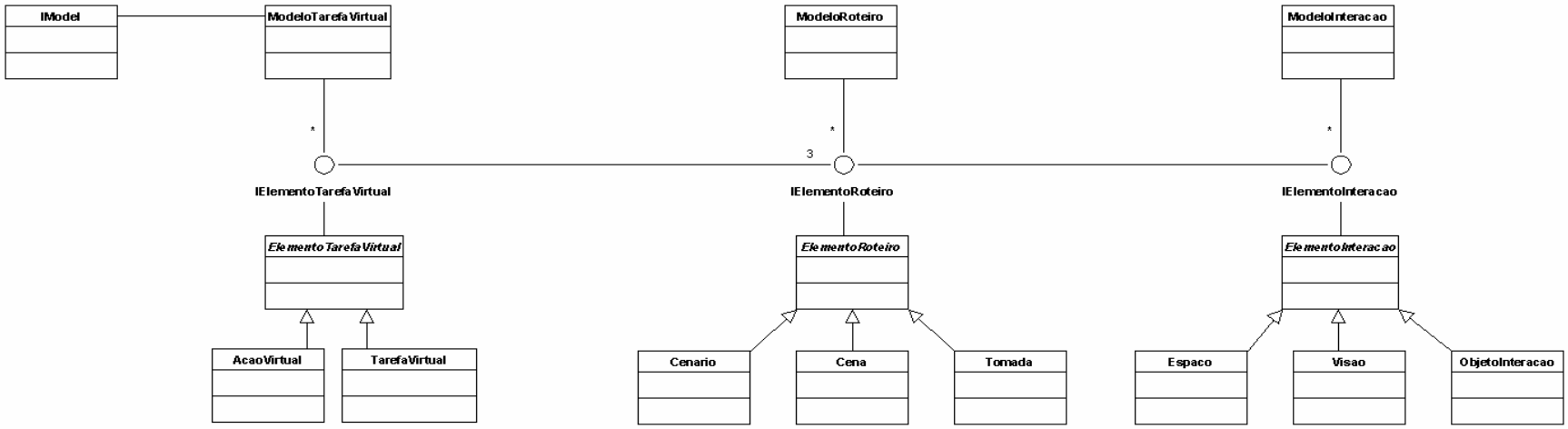


Figura (6.3): Elementos básicos dos Modelos de Tarefa, Roteiro e Interação

Cada um dos modelos supracitados, entretanto, necessitou de uma estrutura de dados para o armazenamento de seus elementos. Todos os modelos foram armazenados em uma árvore específica. O modelo de tarefa virtual na árvore de tarefa virtual, o modelo de roteiro na árvore de roteiro e o modelo da interação na árvore da interação. O módulo TAME já tem a sua estrutura de árvore e maiores detalhes sobre este módulo o leitor deve consultar (Cordeiro, 2003). A Figura (6.4) ilustra as classes base para a implementação dos algoritmos.

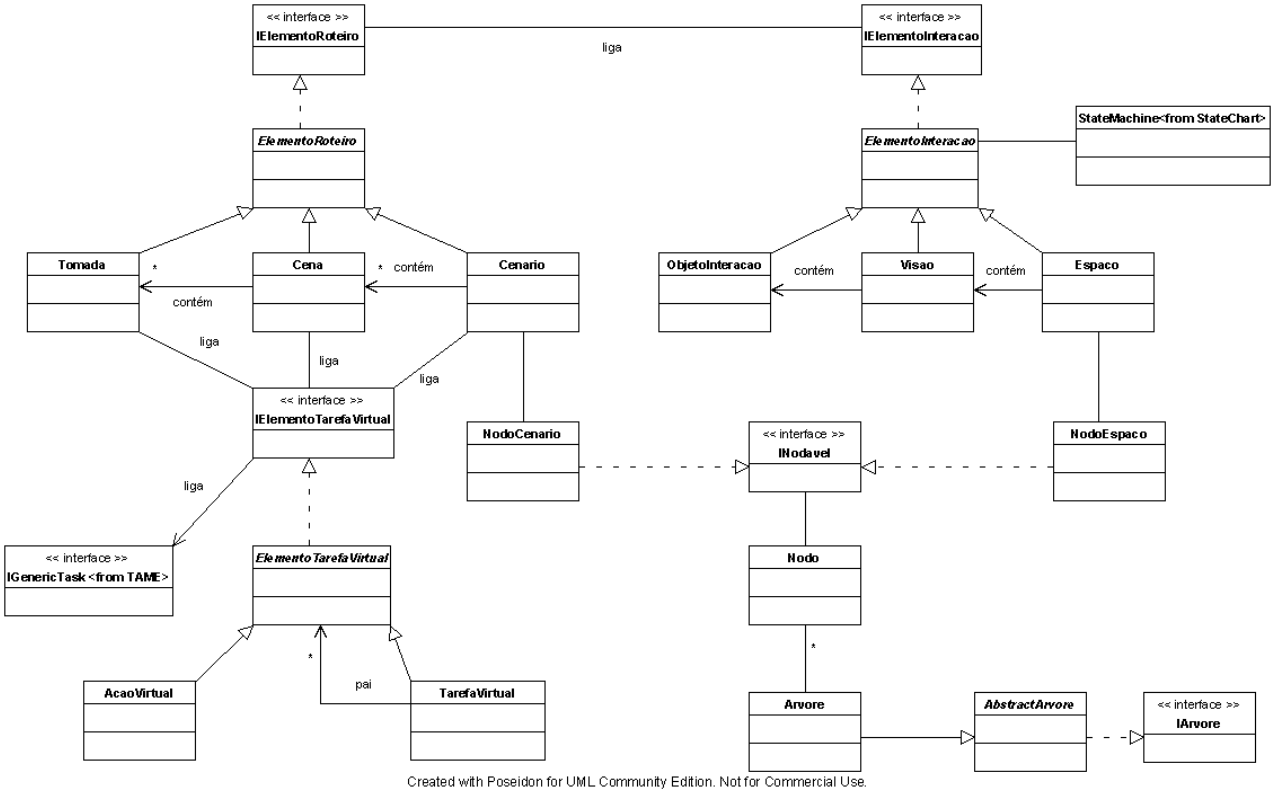


Figura (6.4): Arquitetura base à implementação dos algoritmos

6.3 - Implementação

A linguagem JAVA (JAVA, 2004) foi utilizada para a implementação dos modelos, do componente de diálogo e dos algoritmos. A persistência dos modelos é feita através de arquivos XML e que seguem a definição de um arquivo DTD. A transformação dos arquivos XML nos objetos JAVA é realizada pelo JATO (Krumel, 2001a; Krumel,

2001b; Krumel 2001c). JATO é uma API JAVA e uma linguagem XML de código aberto para transformar documentos XML em um conjunto de objetos JAVA e vice-versa.

6.3.1 - Implementação dos Algoritmos de Obtenção do Modelo da Interação

O modelo da interação é obtido a partir dos algoritmos apresentados no capítulo 5. O primeiro algoritmo necessário para a obtenção do modelo da interação é o algoritmo de obtenção do modelo de roteiro. A partir do modelo de roteiro obtém-se o modelo da interação. Para obter o modelo de roteiro é necessária a classificação dos elementos da tarefa em relação aos elementos do modelo de roteiro. A classificação e a ligação dos elementos da tarefa em elementos do roteiro são feitas através do modelo de tarefa virtual (apresentado na Figura (6.3)). O construtor da classe que representa o modelo de tarefa virtual recebe como parâmetro uma referência à árvore de tarefa do módulo TAME, criando assim o modelo de tarefa virtual e a ligação para o modelo de tarefa real (módulo TAME).

Uma vez obtido o modelo de roteiro, um segundo algoritmo é utilizado para a construção do modelo de interação. Esse algoritmo promove a ligação de elementos do roteiro em elementos da interação. Ao final do processo, toda a estrutura tarefa-interação estará criada e os seus elementos interligados.

6.3.2 - Implementação do Componente de Diálogo EDITOR Estendido

O *Componente Diálogo* do modelo EDITOR Estendido, como apresentado no capítulo 4, é representado pelo formalismo statechart. A implementação deste componente foi feita, também, na linguagem de programação Java e seguiu a especificação UML versão 1.5, apresentada na Figura (6.5). Cada elemento do modelo de interação EDITOR Estendido tem uma referência a uma instância da classe StateMachine (ilustrada na Figura (6.5)). Todos as instâncias dos objetos StateMachine formam o startchart que representa o diálogo da aplicação. Como discutido no capítulo 4, os statecharts são obtidos através dos elementos dos modelos da interação e da tarefa. Cada elemento do modelo da interação será um estado no Statechart, as transições entre estes estados são obtidas através da combinação dos atributos *Método versus Ocorrência* do modelo da tarefa.

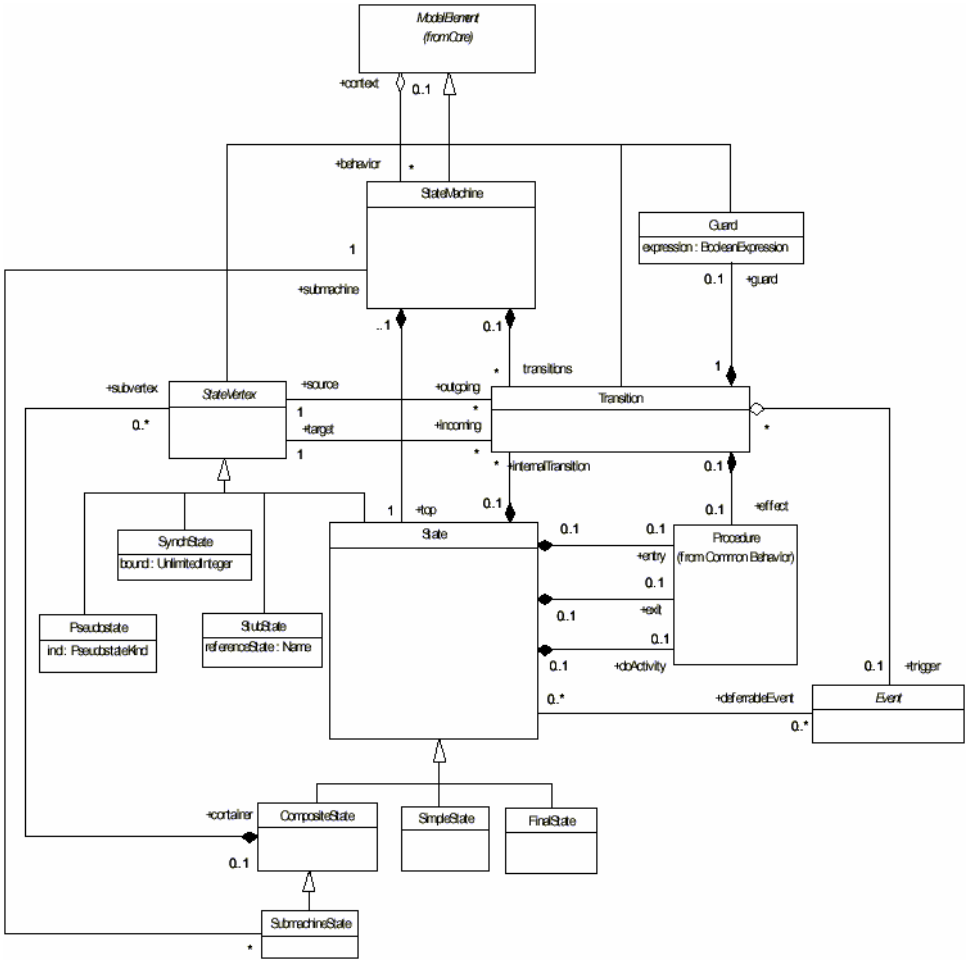


Figura (6.5): Diagrama de classes statecharts da UML v.1.5.

A Figura (6.6) apresenta o diagrama de classes dos eventos permitidos no statechart.

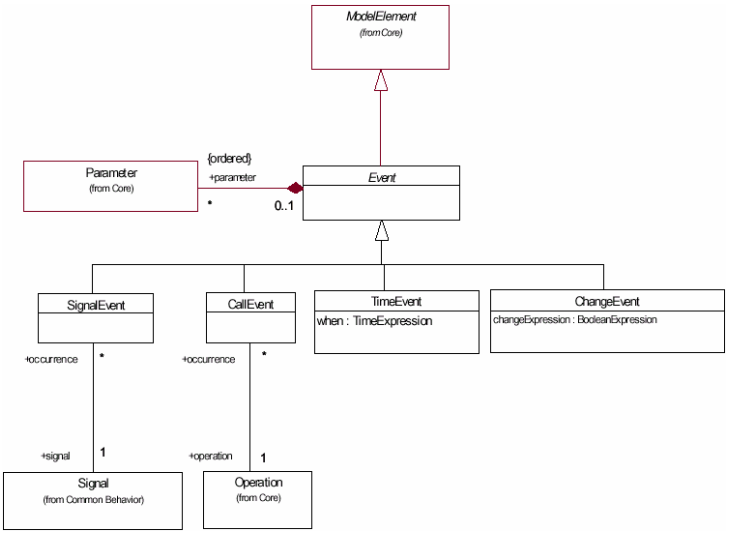


Figura (6.6): Diagrama de classes dos eventos dos statecharts da UML v.1.5

6.3.3 - Implementação dos Mecanismos de Manutenção da Coerência

Os mecanismos de manutenção da coerência estão definidos no capítulo 5 e no anexo B. As operações básicas permitidas ao projetista são:

1. Inclusão
   - 1.1. Incluir *Espaço de Direcionamento* com rearranjo (de outros *Espaços*)
     - 1.1.1. filho do *Espaço Inicial*
     - 1.1.2. filho do *Espaço de Direcionamento*
   - 1.2. Incluir *Espaço de Interação*
     - 1.2.1. filho do *Espaço Inicial*
     - 1.2.2. filho de *Espaço de Interação*
     - 1.2.3. filho de *Espaço de Direcionamento*
   - 1.3. Incluir *Visão (Espaço de Interação)*
   - 1.4. Incluir *Objeto de interação* (*Espaço de Interação*)
2. Exclusão
   - 2.1. Excluir *Espaço de Direcionamento* (sem excluir seus filhos)
   - 2.2. Excluir *Espaço de Interação*
     - 2.2.1. Filho do *Espaço Inicial*
     - 2.2.2. Filho de *Espaço de Direcionamento*
     - 2.2.3. Filho de *Espaço de Interação*
   - 2.3. Excluir *Visão*
   - 2.4. Excluir *Objeto de interação* (se houver link, excluir também o *espaço* associado).

Os algoritmos que implementam as operações acima apresentadas permitem que o projetista promova alterações diretamente no modelo da interação e que estas sejam propagadas de volta até o modelo da tarefa, mantendo a coerência entre os modelos tarefa-interação. Esses mecanismos básicos possibilitarão a construção de uma ferramenta de manipulação visual de elementos do modelo de interação EDITOR Estendido.

6.4 - Conclusão

Este capítulo apresentou as classes e os elementos necessários para a implementação do componente de diálogo, dos algoritmos de obtenção do modelo de roteiro e interação, e dos mecanismos de manutenção da coerência entre os modelos.

A escolha da linguagem JAVA deveu-se ao sucesso de sua utilização no projeto da ferramenta iTAOS. Além disso, JAVA é uma linguagem independente de plataforma e com suporte à linguagem XML, utilizada para a persistência do modelo da tarefa na ferramenta iTAOS. Portanto, nada mais natural que a escolha da linguagem JAVA para a implementação dos artefatos apresentados neste trabalho.

Os artefatos apresentados nesse capítulo serão a base para a construção de uma ferramenta que dê suporte as etapas 2 e 3 da metodologia MEDITE^{+}.

O próximo capítulo apresentará alguns resultados obtidos com a aplicação da metodologia MEDITE^{+}.

Capítulo 7 – Estudo de Caso e Análise de Resultados

Nos capítulos anteriores, foram apresentados os elementos necessários para a preparação do experimento que será apresentado neste capítulo. Será realizada uma avaliação dos resultados alcançados pela utilização da metodologia MEDITE$^+$, utilizando os algoritmos de obtenção do modelo da interação (pelo processo de roteirização) e de manutenção da coerência entre modelos. Além disso, estes resultados serão comparados com os resultados obtidos a partir da utilização de uma metodologia baseada no modelo de tarefas e que utiliza regras de derivação para obtenção do modelo da interação diretamente a partir do modelo da tarefa (Vanderdonckt *et al.*, 2003).

7.1 - Estudo de Caso

Nesta seção, será apresentado um estudo comparativo entre os resultados obtidos a partir da utilização da abordagem preconizada por Vanderdonckt *et al.* e os resultados obtidos a partir da abordagem utilizada por MEDITE$^+$. A descrição do estudo de caso original encontra-se em (Vanderdonckt *et al.*, 2003). O sistema que foi analisado foi o um sistema de pedidos on-line (originalmente *product manegment*). Entretanto, apenas a funcionalidade de pedido por telefone foi desenvolvida. Portanto, o estudo de caso está dividido em três sub-tarefas como descrito abaixo:

- **Identificação do consumidor** – O consumidor liga para uma central e é atendido pela operadora do sistema. Três são os possíveis casos: o consumidor que está ligando ainda não é cadastrado, ou seja, é um novo cliente; o consumidor já é cadastrado e ainda lembra do seu número-código; e, finalmente, o consumidor já é cadastrado, mas não se lembrar do seu número-código. Caso seja um novo consumidor, a operadora pede seu nome, sobrenome e endereço. Caso o consumidor seja um cliente antigo e ainda lembrar do seu número-código no sistema, a operadora insere número-código para fazer a sua identificação. Caso o cliente não se lembre do seu número-código, ele informa o nome e o

sobrenome para que a operadora proceda a sua identificação. Em todos os casos, o consumidor será identificado e será retornado o seu número-código.

- **Preenchimento do pedido** – esta etapa refere-se à realização ao preenchimento do pedido em si. Para cada produto requisitado o consumidor informa o número do produto (ou o seu tipo) e a quantidade desejada. Para cada produto a operadora dita o nome do rótulo do produto ao consumidor para confirmar se o nome do rótulo e o nome do produto desejado coincidem.
- **Confirmação do pedido** – Depois de realizadas a identificação do consumidor e o preenchimento do pedido, a operadora pergunta se qual a forma de pagamento desejada (cartão de crédito, débito ou dinheiro). Por fim a operadora grava o pedido se o consumidor autorizar ou então cancela o pedido.

7.1.1 – CTT + WTN

É apresentada abaixo a modelagem da tarefa como realizada por Vanderdonckt (Vanderdonckt *et al.*, 2003) usando o formalismo CTT (Paterno *et al.*, 1997):

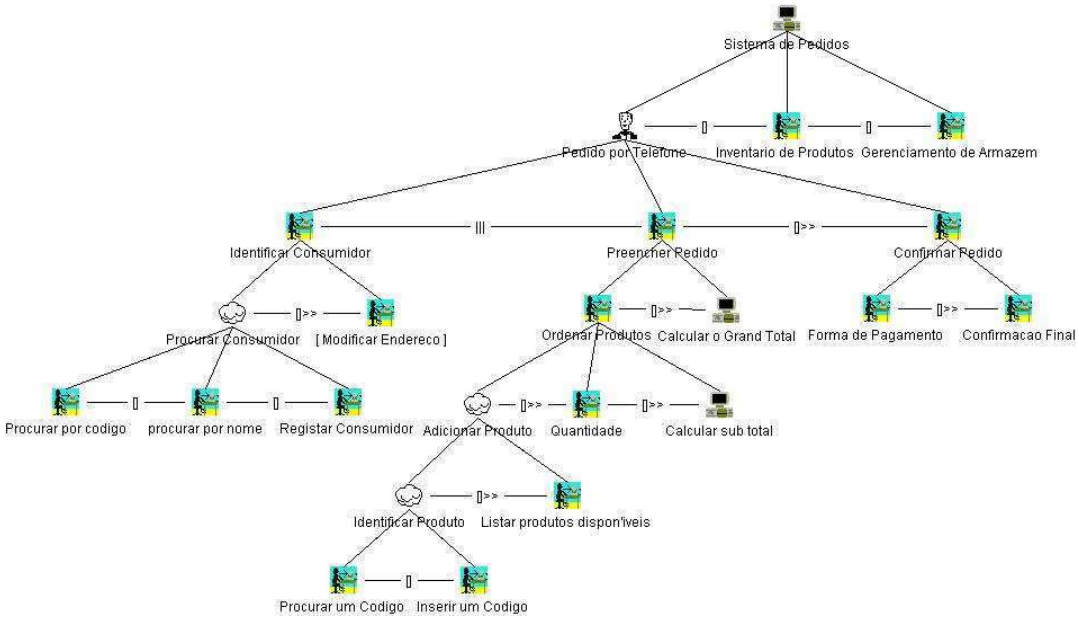


Figura (7.1): Arvore da tarefa CTT do Sistema de Pedidos On-Line (Product Management).

Usando regras de derivação e o formalismo WTN para descrever o diálogo inter-janelas, os autores chegaram aos seguintes protótipos de janelas ilustrados nas Figuras (7.2) a (7.4).

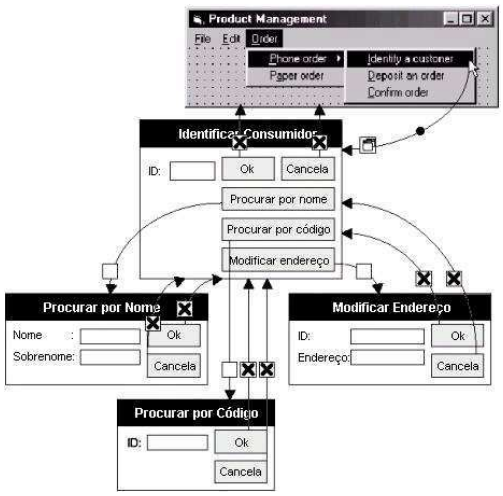


Figura (7.2): Protótipo de janela de identificar o consumidor

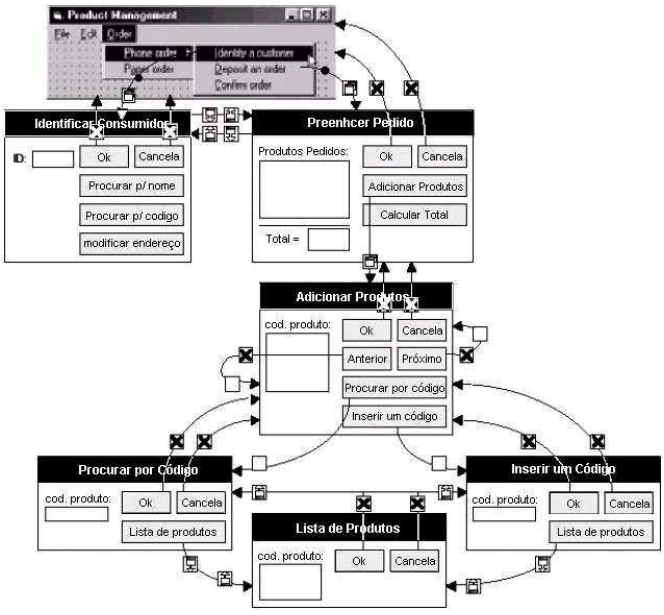


Figura (7.3): Protótipo de janela de preencher o produto

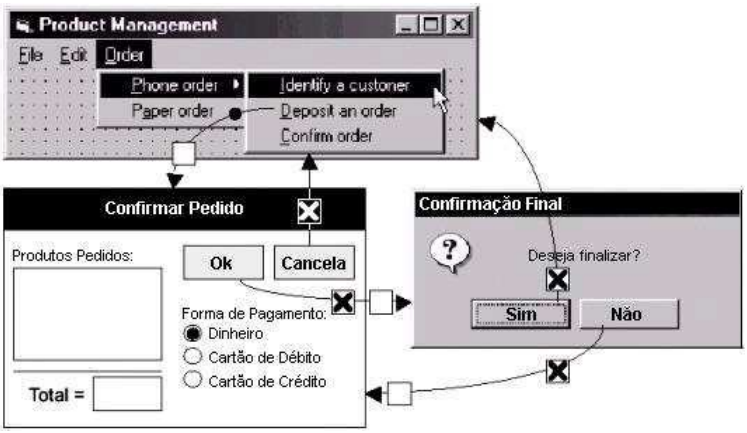


Figura (7.4): Protótipo de janela de confirmar o pedido

7.1.2 – Sistemas de Pedidos On-Line por MEDITE^{+}

MEDITE^{+}, como apresentado no capítulo 5, define um processo de concepção e desenvolvimento de uma interface com as seguintes etapas: (i) análise e modelagem da tarefa (Modelo TAOS); (ii) especificação do modelo de roteiro (Modelo de Roteiro); (iii) especificação conceitual parcial da interação (Modelo EDITOR Estendido); (iv) especificação completa da interação (Modelo EDITOR Estendido); (v) geração do protótipo (OO); e (vi) avaliação.

Segue abaixo a modelagem da tarefa para o mesmo caso, utilizando o formalismo TAOS, correspondendo à primeira etapa da metodologia. Essa modelagem apresenta um maior detalhamento das tarefas em relação à modelagem original. Ou seja, para obtenção dos protótipos de janelas da interface do sistema, utilizando o processo de roteirização, necessitou-se de um maior refinamento das tarefas.

Não foi possível utilizar as regras de transição do modelo de tarefa no modelo de roteiro e do modelo de roteiro no modelo de interação apresentadas no capítulo 5, sem fazer este refinamento. Tal refinamento não implica uma deficiência da metodologia MEDITE^{+}, a análise da tarefa apresentada em (*Vanderdonckt*, 2003) que estava muito pouco detalhada para as necessidades desse trabalho. A Figura (7.5) abaixo ilustra os protótipos que seriam obtidos utilizando a análise da tarefa original.

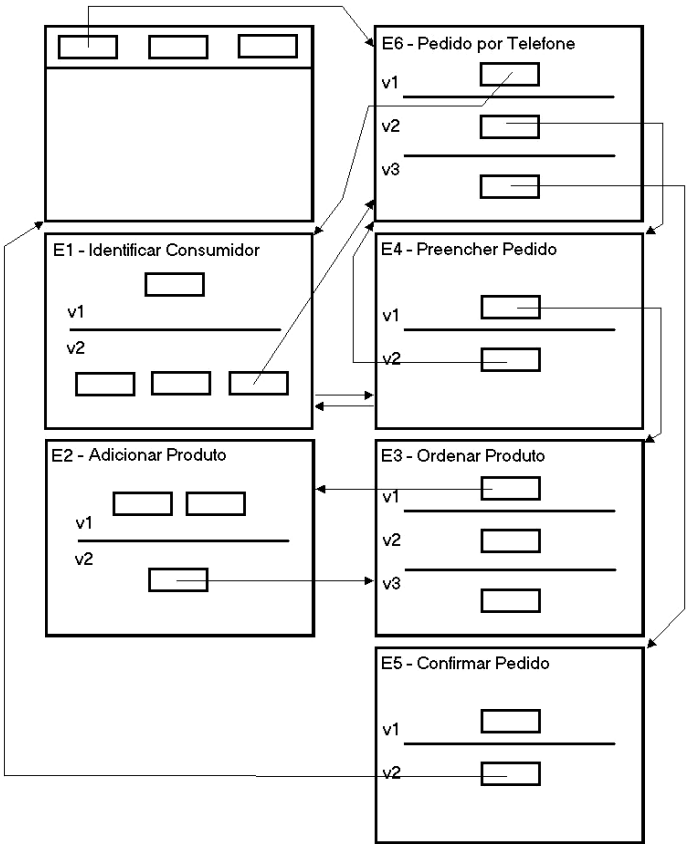


Figura (7.5): Protótipos de janelas utilizando a análise da tarefa original por MEDITE^{+}

Portanto, respeitando a sua versão original e adicionando o detalhamento necessário, a arvore da tarefa TAOS é apresentada nas Figuras (7.6) a (7.9).

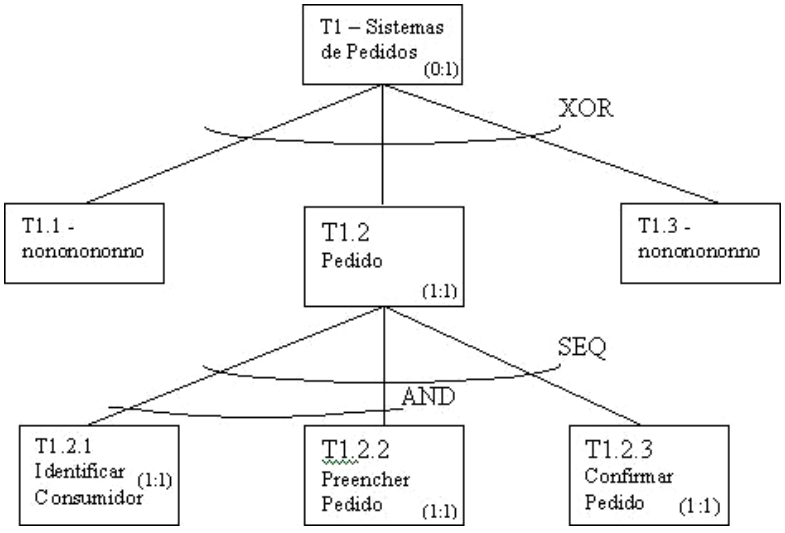
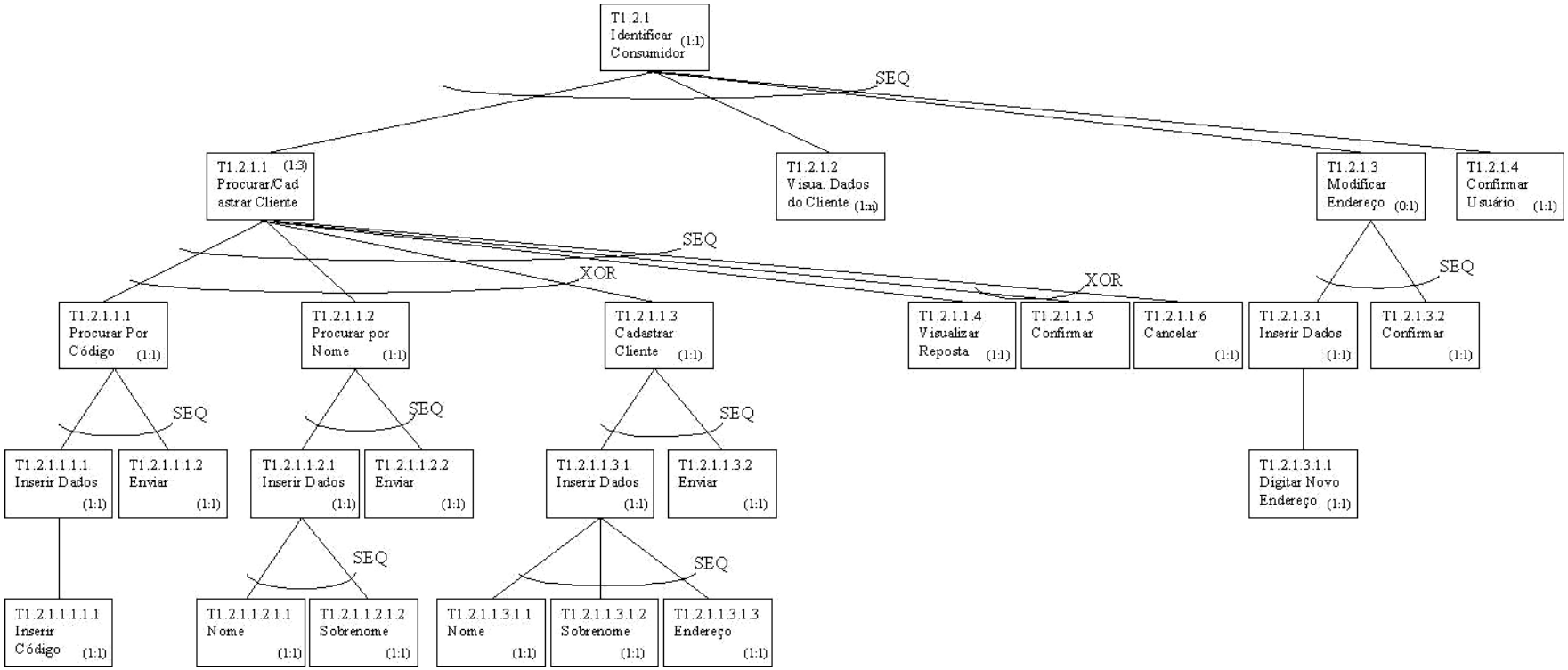


Figura (7.6): Trecho de árvore da tarefa TAOS Sistema de Pedidos On-Line



Figura(7.7): Trecho da árvore da tarefa TAOS Identificar Consumidor

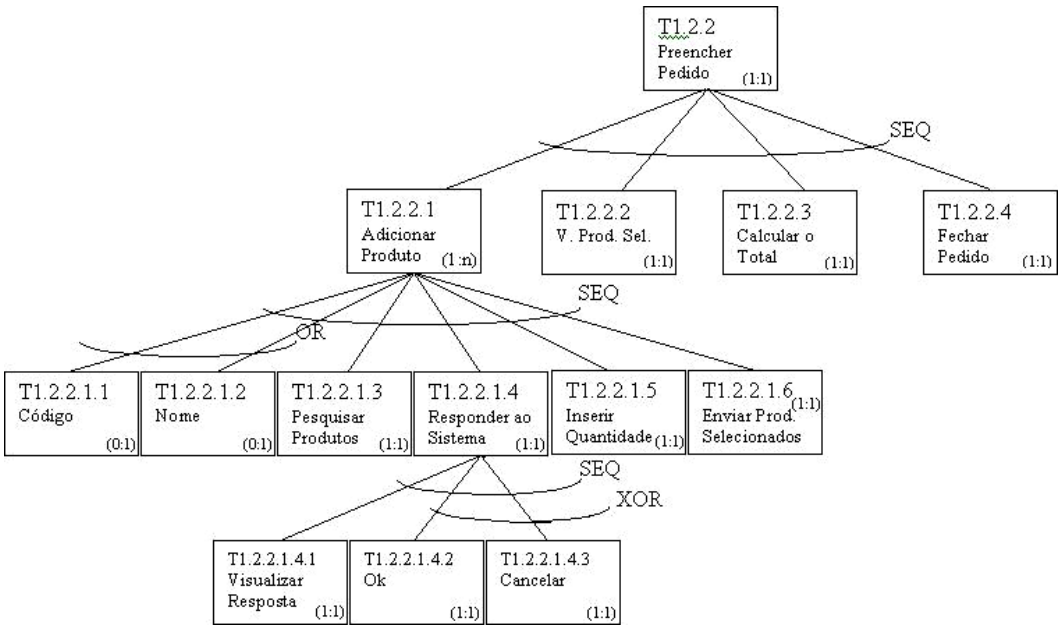


Figura (7.8): Trecho de árvore da tarefa TAOS Preencher Pedido

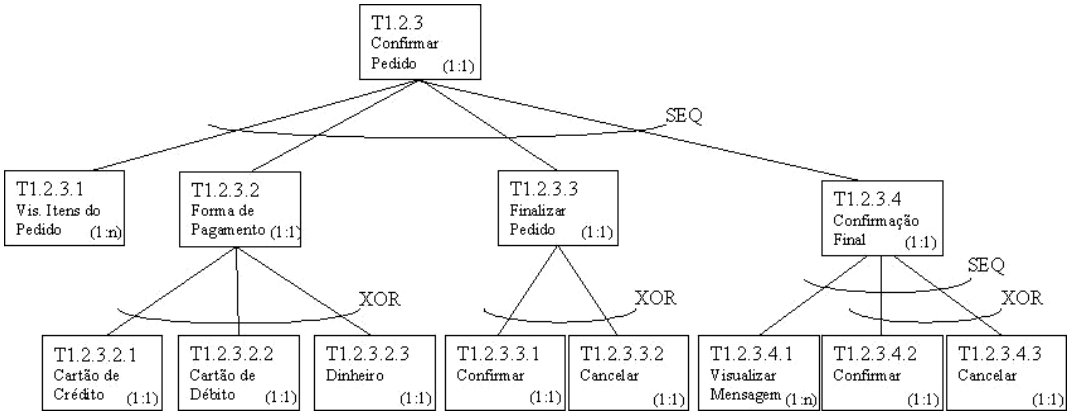


Figura (7.9): Trecho de árvore da tarefa TAOS Confirmar Pedido

O quadro (7.1) apresenta a os detalhes relativos a análise TAOS, a classificação das tarefas em termos dos elementos do roteiro e da interação após a aplicação dos algoritmos de obtenção do modelo da interação apresentados no capítulo 5. A obtenção do modelo de roteiro representa a segunda etapa da metodologia MEDITE$^+$ e a obtenção do modelo parcial da interação EDITOR Estendido representa a terceira etapa da metodologia MEDITE$^+$.

Quadro (7.1): Sistema de Pedidos On-Line TAREFA– ROTERIO – INTERAÇÃO.

Número	Nome	Elemento Roteiro	Elemento Interação
T1	Cons. Sist. de Pedidos On-line	Cenário	Espaço
T1.1	Tarefa qualquer	Cenário	Espaço
T1.2	Pedido por Telefone	Cenário	Espaço
T1.2.1	Identificar Consumidor	Cenário/Cena/Tomada	Espaço/Visão/O.I.
T1.2.1.1	Procurar/Cadastrar Cliente	Cenário/Cena/Tomada	Espaço/Visão/O.I.
T1.2.1.1.1	Procurar por Código	Cenário/Cena/Tomada	Espaço/Visão/O.I.
T1.2.1.1.1.1	Inserir Dados	Cena	Visão
T1.2.1.1.1.1.1	Inserir Código	Tomada	O.I.
T1.2.1.1.1.2	Enviar	Cena/Tomada	Visão/O.I.
T1.2.1.1.2	Procurar por Nome	Cenário/Cena/Tomada	Espaço/Visão/O.I.
T1.2.1.1.2.1	Inserir Dados	Cena	Visão
T1.2.1.1.2.1.1	Nome	Tomada	O.I.
T1.2.1.1.2.1.2	Sobrenome	Tomada	O.I.
T1.2.1.1.2	Enviar	Cena/Tomada	Visão/O.I.
T1.2.1.1.3	Cadastrar Cliente	Cenário/Cena/Tomada	Espaço/Visão/O.I.
T1.2.1.1.3.1	Inserir Dados	Cena	Visão
T1.2.1.1.3.1.1	Nome	Tomada	O.I.
T1.2.1.1.3.1.2	Sobrenome	Tomada	O.I.
T1.2.1.1.3.1.3	Endereço	Tomada	O.I.
T1.2.1.1.3	Enviar	Cena/Tomada	Visão/O.I.
T1.2.1.1.4	Visualizar Resposta	Cena/Tomada	Visão/O.I.
T1.2.1.1.5	Confirmar	Cena/Tomada	Visão/O.I.
T1.2.1.1.6	Cancelar	Cena/Tomada	Visão/O.I.
T1.2.1.2	Visualizar Dados do Cliente	Cena/Tomada	Visão/O.I.
T1.2.1.3	Modificar Endereço	Cenário/Cena/Tomada	Espaço/Visão/O.I.
T1.2.1.3.1	Inserir Dados	Cena	Visão
T1.2.1.3.1.1	Digitar Novo Endereço	Tomada	O.I.
T1.2.1.3.2	Confirmar	Cena/Tomada	Visão/O.I.
T1.2.1.4	Confirmar Usuário	Cena/Tomada	Visão/O.I.
T1.2.2	Preencher Pedido	Cenário/Cena/Tomada	Espaço/Visão/O.I.
T1.2.2.1	Adicionar Produto	Cenário/Cena/Tomada	Espaço/Visão/O.I.
T1.2.2.1.1	Digitar Código do Produto	Cena/Tomada	Visão/O.I.
T1.2.2.1.2	Digitar Nome do Produto	Cena/Tomada	Visão/O.I.
T1.2.2.1.3	Pesquisar Produto	Cena/Tomada	Visão/O.I.
T1.2.2.1.4	Responder ao Sistema	Cena	Visão
T1.2.2.1.4.1	Visualizar Resposta	Tomada	O.I.
T1.2.2.1.4.2	Confirmar item	Tomada	O.I.
T1.2.2.1.4.3	Cancelar item	Tomada	O.I.
T1.2.2.1.5	Inserir Quantidade	Cena/Tomada	Visão/O.I.
T1.2.2.1.6	Lançar Produto	Cena/Tomada	Visão/O.I.
T1.2.2.2	Listar Produtos Lançados	Cena/Tomada	Visão/O.I.
T1.2.2.3	Calcular o Total	Cena/Tomada	Visão/O.I.
T1.2.2.4	Fechar Pedido	Cena/Tomada	Visão/O.I.
T1.2.3	Confirmar Pedido	Cenário/Cena/Tomada	Espaço/Visão/O.I.
T1.2.3.1	Visualizar Itens do Pedido	Cena/Tomada	Visão/O.I.
T1.2.3.2	Forma de Pagamento	Cena	Visão
T1.2.3.2.1	Cartão de Crédito	Tomada	O.I.
T1.2.3.2.2	Cartão de Débito	Tomada	O.I.
T1.2.3.2.3	Dinheiro	Tomada	O.I.
T1.2.3.3	Finalizar Pedido	Cena	Visão
T1.2.3.3.1	Confirmar	Tomada	O.I.
T1.2.3.3.2	Cancelar	Tomada	O.I.
T1.2.3.4	Confirmação Final	Cena	Visão
T1.2.3.4.1	Visualizar Resposta	Tomada	O.I.
T1.2.3.4.2	Confirmar	Tomada	O.I.
T1.2.3.4.3	Cancelar	Tomada	O.I.

As árvores EDITOR, ilustradas nas Figuras (7.10) a (7.19) abaixo, foram obtidas após a aplicação dos algoritmos de obtenção do modelo parcial da interação.

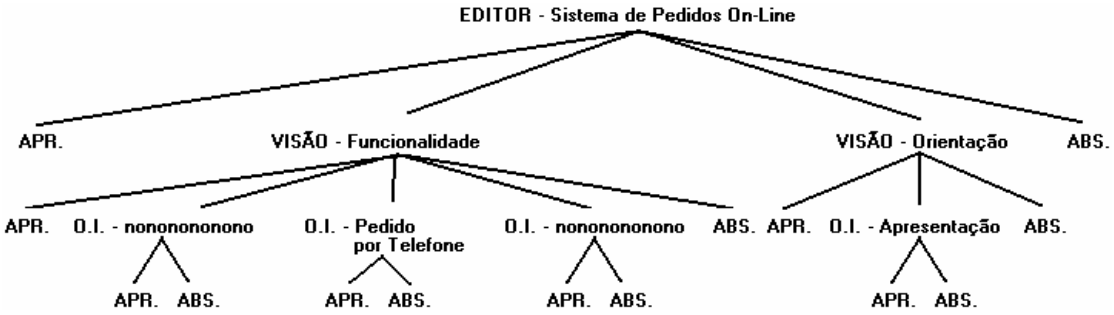


Figura (7.10): Árvore EDITOR Sistema de Pedidos On-Line

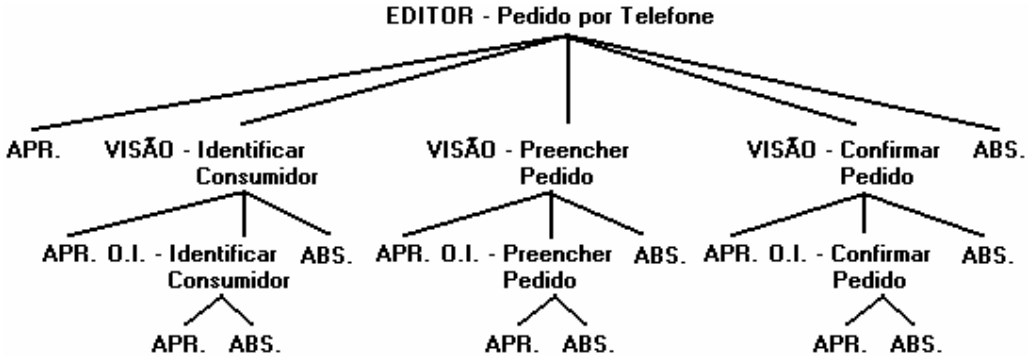


Figura (7.11): Árvore EDITOR Pedido por Telefone

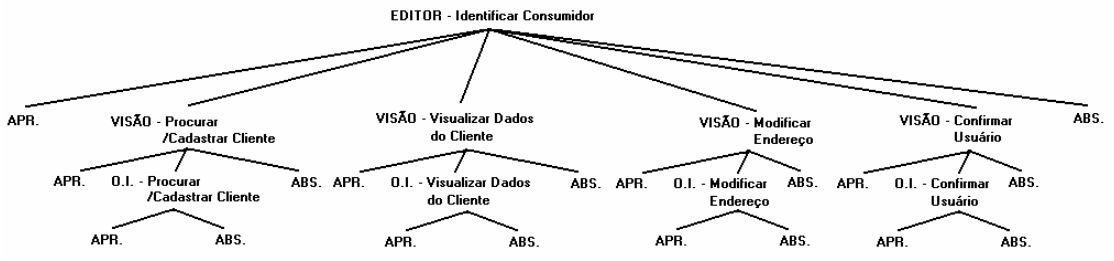


Figura (7.12): Árvore EDITOR Identificar Consumidor

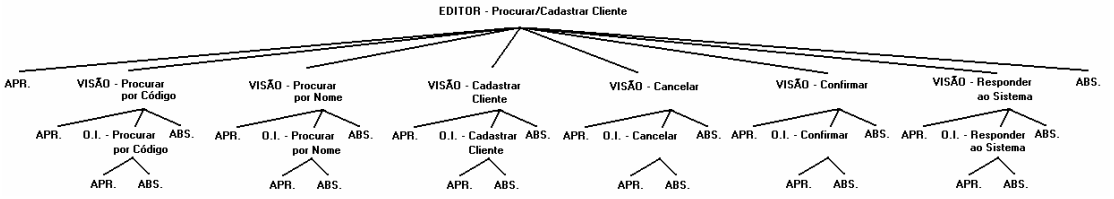


Figura (7.13): Árvore EDITOR Procurar/Cadastrar Cliente

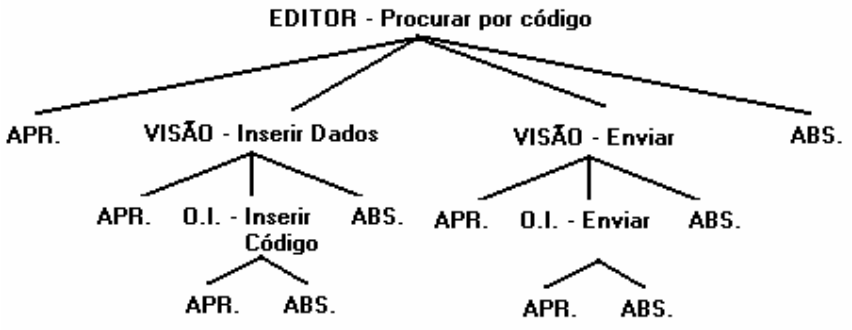


Figura (7.14): Árvore EDITOR Procurar por código

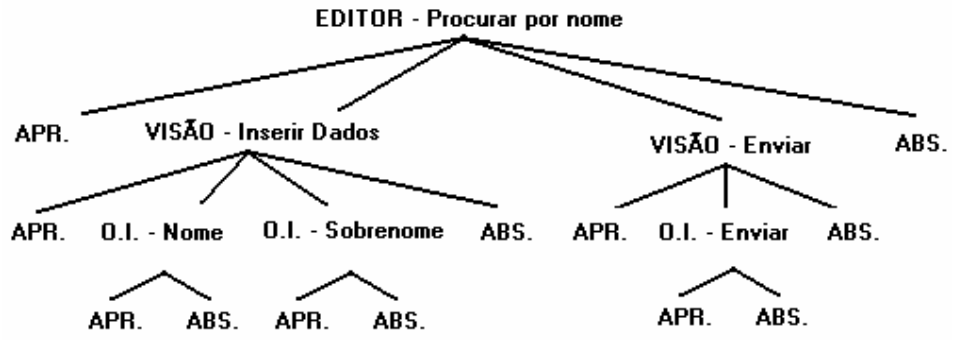


Figura (7.15): Árvore EDITOR Procurar por Nome

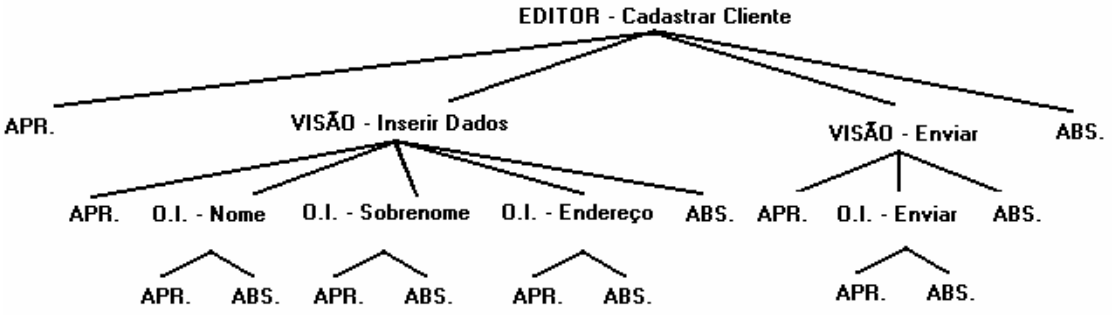


Figura (7.16): Árvore EDITOR Cadastrar Cliente

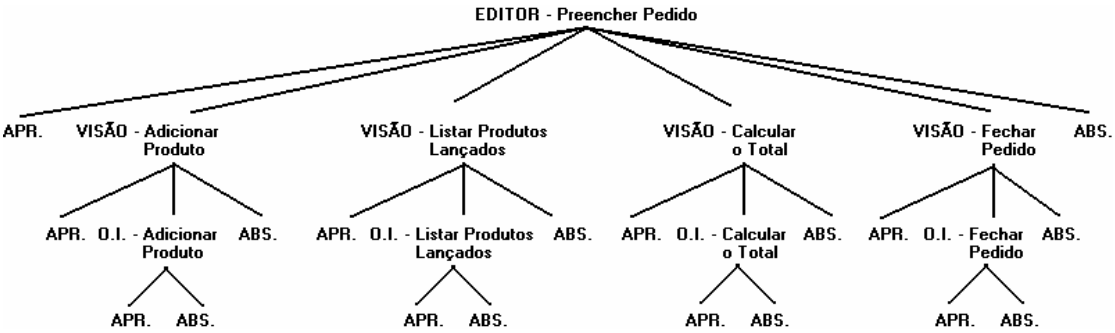


Figura (7.17): Árvore EDITOR Preencher Pedido

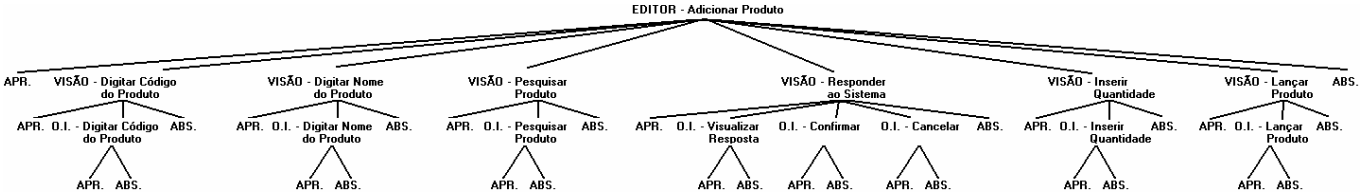


Figura (7.18): Árvore EDITOR Adicionar Produto

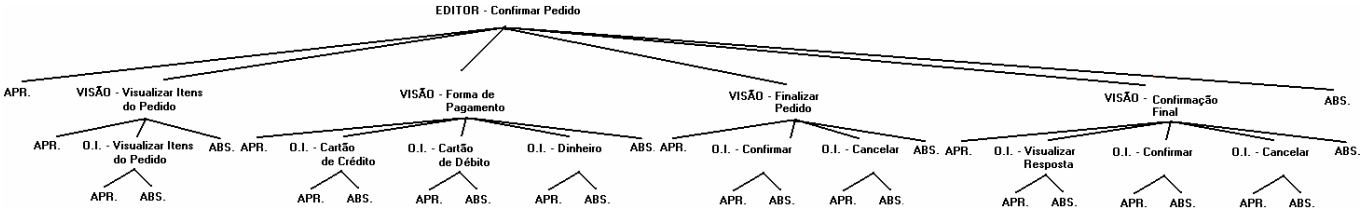


Figura (7.19): Árvore EDITOR Confirmar Produto

Como apresentado no capítulo 4, os statecharts são obtidos através das árvores EDITOR e dos atributos método e ocorrência das tarefas associadas aos elementos do modelo da interação EDITOR Estendido. A Figura (7.20) até a Figura (7.23), ilustram trechos do statechart que representam o diálogo da aplicação que está sendo analisada. Por uma questão de espaço, alguns estados são apresentados nos statecharts de forma minimizada apenas para informar a origem ou o destino dos estados que estão sendo analisados.

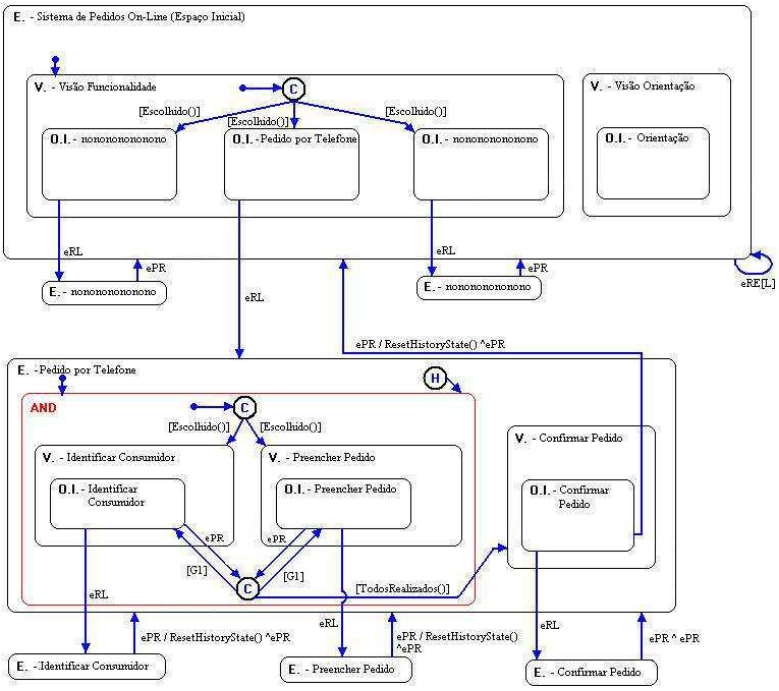


Figura (7.20): Statechart Pedido por Telefone

A Figura (7.20) acima apresenta o *Estado Inicial* (Editor Inicial) do sistema, que contém as *Visões Funcionalidade* e *Orientação.* O estado *E. – Pedido por telefone* é visitado através do método eRL pelo *Objeto de Interação O.I. – Pedido por telefone* da *Visão Funcionalidade*.

A Figura (7.21) abaixo, ilustra o statechart Identificar o Consumidor.

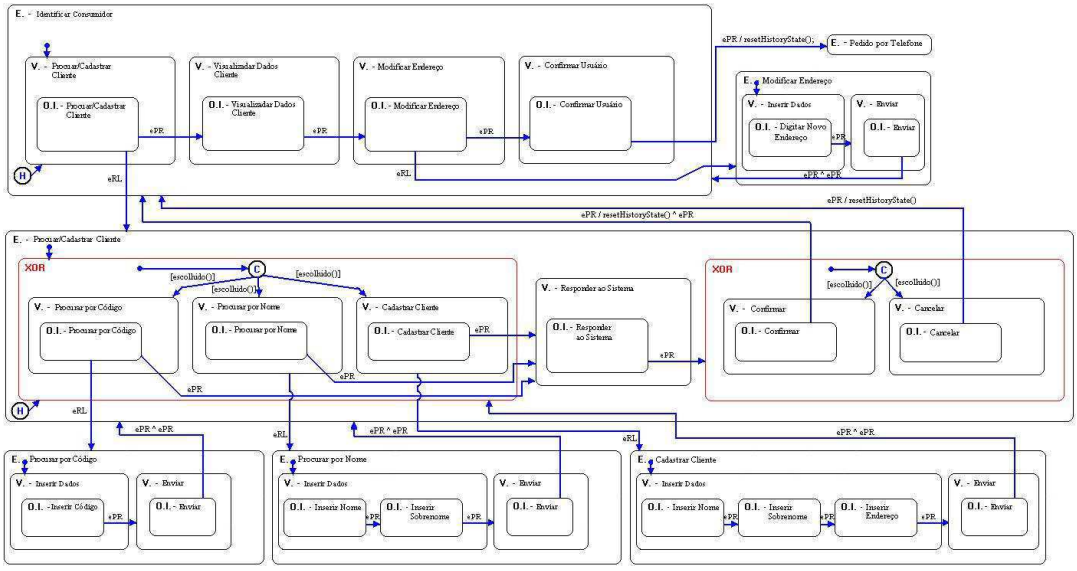


Figura (7.21): Statechart Identificar Consumidor

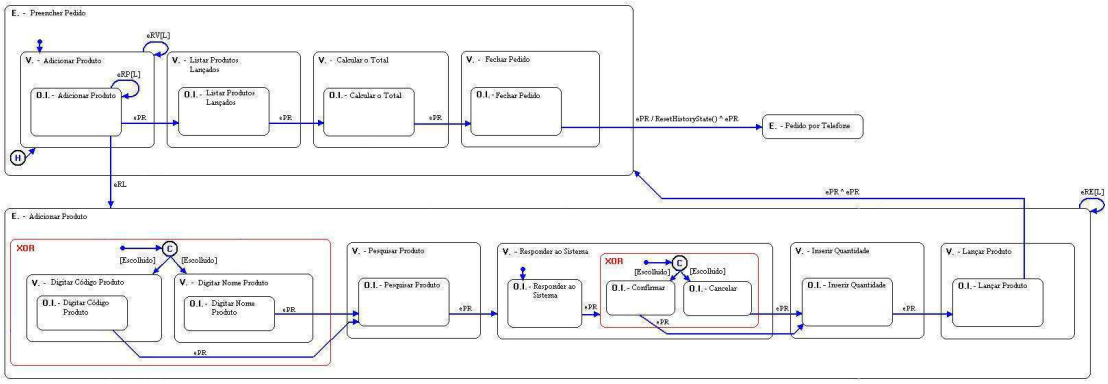


Figura (7.22): Statechart Preencher Pedido

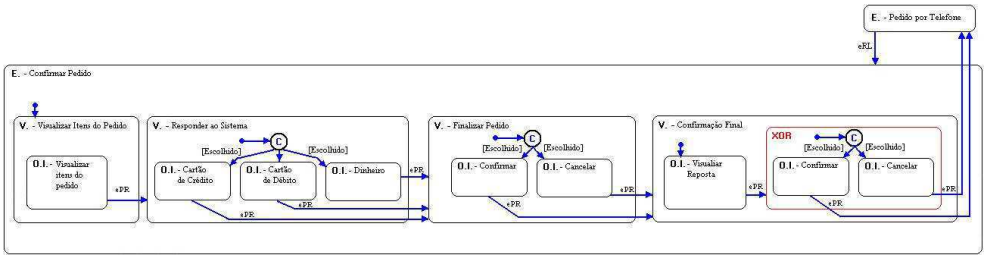


Figura (7.23): Statechart Confirmar Pedido

Como se pode ver, a representação conceitual parcial da interação (representada pelos statecharts), define as janelas e os diálogos inter e intra-janelas. As etapas posteriores da metodologia MEDITE^{+}, as quais não são tratadas nesse trabalho, servirão para identificar apropriadamente (utilizando regras ergonômicas, padrões de interface, etc) os objetos concretos de interação e suas características visuais, como localidade, cor, etc.

Dos statecharts acima construídos, pode-se elaborar facilmente esboços das janelas do sistema e de seus diálogos associados, como apresentado nas Figuras (7.24) a (7.26).

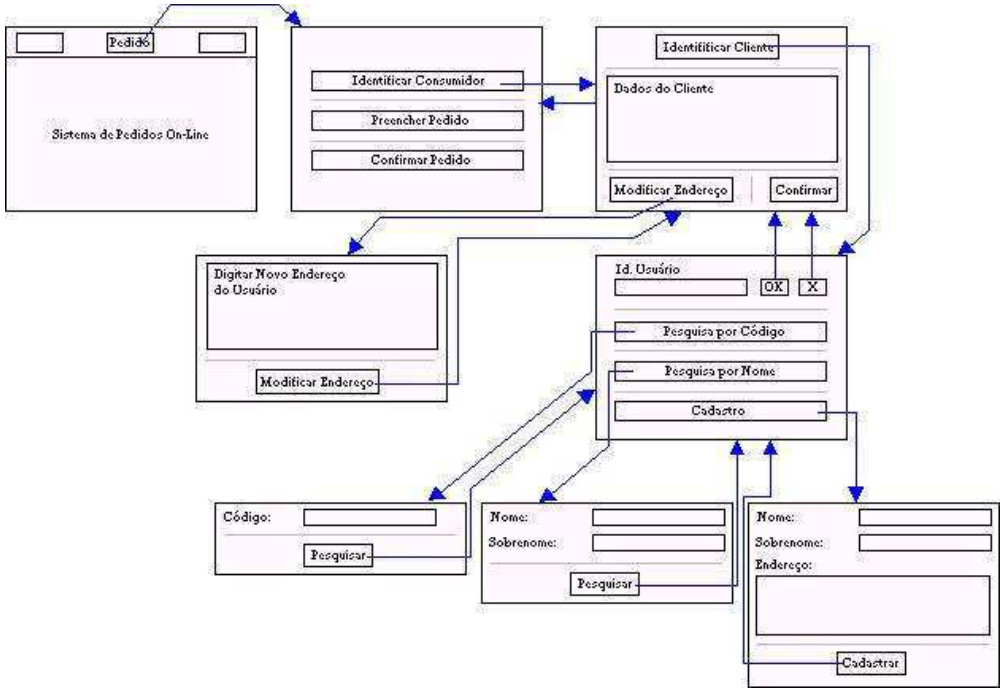


Figura (7.24): Protótipos de Identificar Consumidor

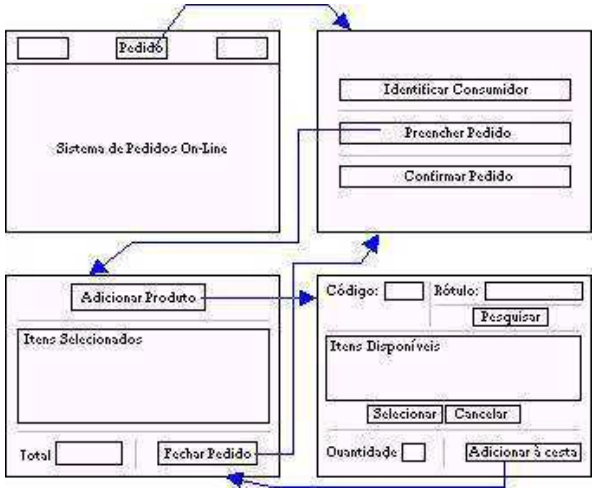


Figura (7.25): Protótipos de Preencher Pedido

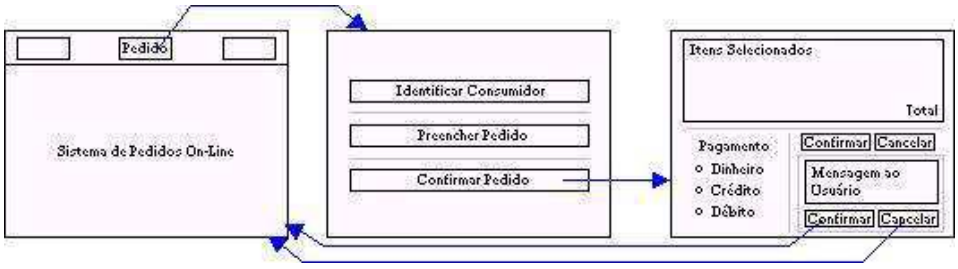


Figura (7.26): Protótipos de Confirmar Pedido

7.2 - Análise Comparativa

A especificação conceitual parcial da interação obtida a partir da utilização de MEDITE$^+$ para o estudo de caso considerado é semelhante ao modelo da interação apresentado em (Vanderdonckt *et al.*, 2003) para o mesmo caso. Para os trechos de tarefa *identificar consumidor* e *preencher pedido* os protótipos das janelas (apresentação) obtidos utilizando MEDITE$^+$ foram iguais aos obtidos por Vanderdonckt. Já o trecho de tarefa *preencher pedido*, a utilização de MEDITE$^+$ resultou em um menor número de janelas para a realização deste trecho de tarefa. Enquanto em (Vanderdonckt *et al.*, 2003) a operadora necessitaria de quatro janelas para realizar esta tarefa, utilizando MEDITE$^+$, a operadora necessitaria de apenas duas janelas, fazendo a tarefa de maneira aparentemente mais simples.

Quanto ao diálogo da aplicação, (Vanderdonckt *et al.*, 2003) utiliza o formalismo WTN para descrever o diálogo inter-janelas através de operações de abertura, fechamento, maximização, entre outras operações de transições entre janelas. Esse formalismo, entretanto, não contempla o diálogo interno a uma janela (diálogo intra-janela) o que exigiria a utilização de outro(s) formalismo(s) como mencionado em (Vanderdonckt *et al.*, 2002).

Neste trabalho, o formalismo preconizado para a representação do diálogo é o formalismo statechart, que, além de permitir a representação tanto do diálogo inter quanto intra-janelas, possui diversas outras vantagens como, fazer parte de uma linguagem universal - UML, possuir diversas ferramentas disponíveis, possuir uma especificação formal além de todas as demais vantagens apresentadas no capítulo 4 .

O modelo da interação para o caso em estudo, obtido na formulação original, através da aplicação de regras de derivação, conforme mencionado no capítulo 2, exige uma interpretação de dados armazenados no modelo da tarefa. Além disso, não há indicação de uma ferramenta efetiva disponível para auxiliar o projetista nessa transformação. Ao contrário, a abordagem implementada em MEDITE$^+$ permite a automatização do processo de transformação do modelo da tarefa no modelo da interação, liberando o projetista do esforço de interpretação e de eventuais erros.

Além das vantagens acima citadas, a abordagem implementada por MEDITE$^+$ permite a rastreabilidade entre elementos de modelos distintos. Tal característica, como apresentado no capítulo 5, permitiu a construção de mecanismos de manutenção da coerência entre os modelos. Portanto, com a introdução da rastreabilidade é possível

realizar alterações diretamente no modelo de interação e propagar essas mudanças até o modelo da tarefa, mantendo a coerência entre os modelos, ou seja, os mecanismos de manutenção da coerência baseados na rastreabilidade propagam de volta as alterações, automaticamente, até o modelo da tarefa, evitando a remodelagem manual da tarefa do usuário e todos as conseqüências indesejáveis que a modelagem manual pode causar.

7.3 - Conclusão

Este capítulo apresentou um estudo comparativo entre os resultados a abordagem preconizada por Vanderdonckt (Vanderdonckt *et al.*, 2003) e a utilizada por MEDITE$^+$. Entre os resultados Os protótipos conseguidos com a utilização de MEDITE$^+$ foram bem semelhantes aos obtidos por Vanderdonckt, com a vantagem da utilização de MEDITE$^+$ apresentar o diálogo interno e externo as janelas, permitir a obtenção automática do modelo da interação, e garantir a manutenção automática da coerência entre os modelos envolvidos através dos mecanismos de manutenção da coerência, evitando o re-projeto manual (pelo projetista) dos modelos utilizados em etapas anteriores à modelagem da interação. Os mecanismos propostos, entretanto, ainda precisam ser utilizados para por um ambiente gráfico para que possa permitir a manipulação efetiva dos elementos do modelo da interação.

No próximo capítulo, serão apresentadas as conclusões gerais desse trabalho assim como as sugestões de trabalhos que promovam a sua continuidade.

Capítulo 8 - Conclusão

8.1 - Conclusões e Resultados

O objetivo geral deste trabalho de dissertação foi definir uma variante da metodologia de concepção e desenvolvimento de interfaces homem-computador MEDITE como pela introdução do processo de roteirização para a obtenção do modelo da interação – MEDITE$^+$. A introdução dessa nova abordagem possibilitou (i) a automatização do processo de obtenção de uma especificação conceitual parcial da interação a partir da modelagem da tarefa, e (ii) a manutenção automática da coerência entre os modelos da tarefa e da interação.

A introdução da nova abordagem em MEDITE implicou a divisão do processo de obtenção da especificação conceitual parcial da interação em dois sub processos, a saber: geração do modelo de roteiro a partir do modelo de tarefa; e geração da especificação conceitual parcial da interação a partir do modelo de roteiro. Para a introdução dessa divisão foram necessárias a extensão do modelo de interação EDITOR (modelo EDITOR Estendido) e a definição de um formalismo (statechart) para a sua representação.

A introdução do processo de roteirização aliada à definição formal (explicita) do componente de diálogo do modelo da interação permitiu a definição de um algoritmo para a obtenção automática deste componente (componente de diálogo) a partir (i) dos elementos do modelo da tarefa e (ii) dos elementos do componente de apresentação do modelo da interação.

MEDITE$^+$ – foi utilizada com sucesso em trabalhos de cunho teórico, como mostrado no capítulo 7, e em trabalhos de desenvolvimento efetivo de interfaces para aplicações concretas. Entre esses, pode-se citar os realizados por alunos matriculados na disciplina Interface Homem-Máquina (a qual o autor foi estagiário docente) oferecida pelo Departamento de Sistemas e Computação (DSC) da UFCG no período 2004.1, quais sejam:

8.1.1 - SOBI, Sistema Operacional de Bibliotecas

SOBI, Sistema Operacional de Bibliotecas. A interface (gráfica) com o usuário desse sistema havia sido projetada e implementada de maneira *ad. hoc*., sem a utilização de nenhuma metodologia específica de concepção e desenvolvimento de interfaces homem-computador. Houve, portanto, um re-projeto da interface gráfica utilizando MEDITE^{+}, o que, no dizer do projetista "muito contribuiu para a evolução da interface do sistema no que diz respeito à coerência, consistência e usabilidade". O sistema está atualmente sendo utilizado, com sucesso, na biblioteca setorial do DSC (mini-blio).

Pôde-se constatar neste estudo de caso a grande evolução da interface gráfica com o usuário proporcionada pela utilização da metodologia MEDITE^{+}. Este estudo re-projetou todas as telas do sistema, levando-se me consideração à análise da tarefa do usuário e suas características. Ao final deste projeto constatou-se o incremento da usabilidade do sistema, reduzindo os erros em sua operação e, conseqüentemente, otimizando o trabalho.

8.1.2 – COOKBOOK

COOKBOOK, um "livro de receitas" *stand-alone* para a consulta de métodos de avaliação de usabilidade. O objetivo deste sistema é classificar ou agrupar esses métodos a fim de apresentá-los de uma forma resumida, ajudando leitores principiantes ou mesmo engenheiros de software a obter um guia prático com informações básicas sobre estes. O projeto da interface do COOKBOOK foi totalmente elaborado utilizando a metodologia MEDITE^{+}. Na avaliação do projetista, "a adoção da metodologia MEDITE^{+} realizou um papel importantíssimo no projeto do sistema, pois os aspectos de usabilidade da interface foram resolvidos de forma mais rápida, sobrando tempo para o projetista desenvolver sua criatividade na concepção dos elementos gráficos e de interação, além de ajudar bastante na prototipagem, pois a construção do modelo de interação (base para construir os protótipos) se mostrou bastante simples".

O projetista pode seguir todas as etapas preconizadas na metodologia MEDITE^{+}, realizando um projeto efetivamente centrado no usuário. O projetista não possuía nenhum

conhecimento prévio sobre projeto e desenvolvimento de interfaces-homem máquina. Apenas seguindo os passos da metodologia, foi possível a construção de telas usáveis e que levavam em conta aspectos relativos aos usuários.

8.1.3 – Sistema Comercial de Vídeo Locadora

Um Sistema comercial de vídeo locadora, onde o usuário pode fazer consultas, cadastros e empréstimos em geral, ou seja, um sistema comercial completo para automação de vídeo locadoras. Segundo o projetista, "apenas seguindo a metodologia à risca, foi possível chegar a um protótipo de tela muito próximo do que o cliente desejava. Ainda, a metodologia conduziu à criação de telas limpas. Com pouco conhecimento prévio sobre características de boas interfaces homem-máquina, foi possível chegar a um resultado em que temos telas com poucos componentes, claras de serem entendidas e, principalmente, intuitivas. Dentro deste aspecto a metodologia cumpriu seu papel, levando um projetista de interfaces semi-leigo a alcançar resultados muito bons, mesmo sem experiência prévia".

Mais um caso de projetista que construía as interfaces de maneira *ad. hoc*. A utilização da metodologia auxiliou não só a construção da interface, mas também, o levantamento de requisitos para o sistema. Assim como as demais, o projeto foi totalmente centrado no usuário com resultados muito bons.

Pôde ser constatado que a idéia de projeto e desenvolvimento de interfaces foi bastante modificada após a utilização da metodologia. Os projetistas reconheceram que o desenvolvimento centrado no usuário facilitou o desenvolvimento do sistema e reduziu o número de alterações na interface em si.

Como se constata dos casos supracitados, MEDITE^{+} mostrou-se muito eficaz e ajudou, com sucesso, os projetistas na obtenção de uma especificação da interação a partir do modelo da tarefa que lhes permitiu construir facilmente telas intuitivas, fáceis de serem entendidas e manipuladas, e consistentes. Entretanto, uma crítica unânime à utilização da metodologia foi a necessidade de construção manual de todos os artefatos exigidos, o que consumiu uma considerável parcela de tempo de trabalho.

Este trabalho de dissertação vem ao encontro de uma solução para o principal problema citado pelos projetistas sobre a utilização da metodologia. A definição e implementação (i) dos algoritmos de obtenção automática do modelo de interação, (ii) dos mecanismos de manutenção da coerência entre os modelos e (iii) do componente de diálogo, através de sua representação statechart servirão de base para a construção de uma ferramenta que auxilie o projetista na realização das etapas 2 e 3 da metodologia MEDITE^{+}, gerando automaticamente os artefatos, e permitindo que o projetista realize mudanças no modelo da interação mantendo automaticamente a coerência com o modelo da tarefa.

O trabalho do Grupo de Interfaces Homem-Máquina (GIHM), portanto, não se extingue com a conclusão deste trabalho. Ainda são necessários alguns estudos para desenvolver as demais etapas da nova metodologia e promover um suporte ferramental completo para as suas etapas. A próxima seção apresenta sugestões para trabalhos futuros.

8.2 – Trabalho Futuros

Para dar prosseguimento ao conjunto de necessidades para o desenvolvimento da metodologia MEDITE^{+} são sugeridos os seguintes trabalhos:

8.2.1 – Implementação de uma Ferramenta que utilize as Funcionalidades Desenvolvidas

É preciso implementar uma ferramenta que possua uma interface gráfica para a manipulação do modelo de interação EDITOR Estendido. Esta ferramenta utilizará as classes e os algoritmos de obtenção do modelo da interação e os mecanismos de manutenção da coerência entre os modelos definidos neste trabalho. A implementação desta ferramenta deve permitir aos projetistas a manipulação dos elementos do modelo da interação e/ou do componente de diálogo da aplicação. Esta ferramenta deve receber como entrada um arquivo XML gerado pela ferramenta iTAOS, produzindo uma representação gráfica (esboço) editável do modelo da interação gerado pelo processo de roteirização. Os mecanismos de manutenção da coerência entre os modelos apresentados nesse trabalho de dissertação são mecanismos elementares. É necessário analisar todas as operações que possam ser realizadas no modelo da interação, utilizando os mecanismos básicos para

compor mecanismos de manutenção de coerência de mais alto nível que possam ser disponibilizados para manipulação em uma interface gráfica. Além disso, o componente de diálogo apresentado no capítulo 4 deste trabalho, que utiliza o formalismo statechart, leva em consideração eventos genéricos. Entretanto, é necessário estender cada um dos eventos apresentados no capítulo para os objetos de interação de uma *toolkit* adotada. Essa atividade deve ser realizada no momento da especificação conceitual completa da interação.

8.2.2 – Desenvolvimento das Demais Etapas da Metodologia MEDITE$^+$

As modificações realizadas neste trabalho disseram respeito apenas à fase da obtenção de uma especificação conceitual parcial da interação da antiga metodologia MEDITE. A obtenção da especificação conceitual completa da interação nos projetos realizados com MEDITE$^+$, foi baseada apenas no conhecimento dos projetistas e em heurísticas de projeto. Entretanto, é necessário desenvolver as demais etapas da metodologia. É necessário realizar um estudo sobre os conhecimentos envolvidos no momento da especificação completa da interação, também conhecida como "arte final". Estudos sobre padrões de construção de interfaces e conhecimento ergonômico são importantes para esta fase.

8.2.3 – Definição dos Mecanismos de Avaliação (dos Pontos da Metodologia)

Por fim, é necessário um estudo mais elaborado dos mecanismos de avaliação presentes em cada etapa da metodologia. Definições formais, assim como auxílio ferramental, são importantes para dar suporte a estas avaliações.

Referências Bibliográficas

(Barbosa et al., 2002) BARBOSA, S. D. J., SOUZA, C. S. de, PAULA, M. G. de, SILVEIRA, M. S., "Modelo de Interação como Ponte entre o Modelo de Tarefas e a Especificação da Interface", IHC'2002 Fortaleza, 2002.

(Bodart e Vanderdonckt, 1993) BODART, F., VANDERDONCKT, J., "Guide Ergonomique de la presentation des applications hautement interactives", Namur, Belgique: Presses Universitaires de Namur, 1993.

(Bodart et al., 1994) BODART, F., HENNEBERT, A.-M., LEHEUREUX, J.-M., PROVOT, I., VANDERDONCKT, J., "A Model-based Approach to Presentation: A Continuum from Task Analysis to Prototype", in Proc. of 1st Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'94 (Bocca di Magra, 8-10 juin 1994), F. Paternó (éd.), Eurographics Series, Berlin, 1994, pp. 25-39.

(Bodart et al., 1995) BODART, F., HENNEBERT, A.-M., LEHEUREUX, J.-M., PROVOT, I., SACRÉ, B., VANDERDONCKT, J., "Towards a Systematic Building of Software Architectures: the TRIDENT methodological guide", in Proc. of 2nd Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'95 (Toulouse, 7-9 juin 1995), Ph. Palanque & R. Bastide (eds.), Springer-Verlag, Vienne, 1995, pp. 262-278.

(Brochot, 1990) BROCHOT, J. N., "La manipulation direct en interface homme-machine", Thèse de Doctorat, Université des Sciences et Techniques du Languedoc, Décembre, 1990.

(Card et al., 1983) CARD, S. K., MORAN, T. P., NEWELL, A., "The psychology of human-computer interaction". Lawrence Erlbaum Associates, Hillsdale (1983).

(Cassandras e Lafortune, 1999) CASSANDRAS, C. G., LAFORTUNE, S. "Introduction to Discrete Event Systems". Ed. Kluwer Academic Publishers, USA, 1999.

(Clerckx *et al.*, 2004) CLERCKX, T., LUYTEN, K., CONINX, K., "The mapping problem back and forth: customizing dynamic models while preserving consistency" in Proceedings of the 3rd annual conference on Task models and diagrams 2004, Prague, Czech Republic November 33 - 42, Pages: 113 - 120, 2004.

(Cordeiro, 2003) CORDEIRO, P. B., "Projeto e implementação do módulo TAME da ferramenta iTAOS para análise e modelagem da tarefa",

Dissertação de Mestrado - COPIN, UFCG, Campina Grande PB, Fevereiro de 2003.

(Coutaz, 1987) COUTAZ, J. *The Construction of User Interfaces and the Object Paradigm*, The European Conference on Object Oriented Programming, ECOOP'87, Paris, França, 1987.

(Coutaz, 1990) COUTAZ, J. *Interfaces Homme-Ordinateur – Conception et Réalisation*, Bordas, Paris, 1990.

(Coutaz e Bass, 1991) COUTAZ, J., BASS, L., "Developing software for the user interface", SEI Series in software engineering, Addison-Wesley publishing company, 1991.

(Dodani *et al.*, 1989) DODANI et. al., "Separation of Powers", Byte, mars 1989.

(Douglass, 2004) UML Statecharts, Bruce Powel Douglass, disponível em www-md.e-technik.uni-rostock.de/ ma/gol/ilogix/umlsct.pdf acessado em 10/09/2004

(Elwert e Schlungbaum, 1995) ELWERT, T., SCHLUNGBAUM, E., "Modelling and Generation of Graphical User Interfaces in the TADEUS Approach". In Designing, Specication and Verication of Interactive Systems, pages 193{208, Vienna, 1995. Springer.

(Elwert *et al.*, 1996) ELWERT, T., SCHLUNGBAUM, E. "Dialogue Graphs – A Formal and Visual Specification Technique for Dialogues Modeling". In Proceedings of the BCS-FACS Workshop pn Formal Aspects of the Humam Computer Inteface, Sheffield Hallam University, 10-12 September 1996.

(Eshuis e Wieringa, 2000) ESHUIS, R., WIERINGA, R., "Requirements-Level Semantics For UML Statecharts" in Formal Methods for Open Object-Based Distributed Systems IV - Proc. FMOODS'2000, September, 2000, Stanford, California, USA

(Fedeli *et al.*, 2003) FEDELI, R. D., POLLONI, E.G. F., PERES, F. E., "Introdução à Ciência da Computação", São Paulo, Thomson Learning, 2003

(Figueiredo e Braga, 1997) FIGUEIREDO, J. C. A., BRAGA, J. D. M., "Curso de Redes de Petri". I Seminário Regional dos Estudantes de Engenharia do Nordeste, UFPB, Campina Grande, PB, Abril, 1997.

(Fowler e Scott, 2000) FOWLER, M., SCOTT, K., "UML essencial: um breve guia para a linguagem - padrão de modelagem de objetos". Tradução Vera Pezerico e Christian Thomas Price. 2a. Edição. Porto Alegre,

Bookman, 2000

(Furtado, 1997) FURTADO, M. E. S., “Mise en oeuvre d’une méthode de conception d’interfaces adaptatives pour des systèmes de supervision à partir des spécification conceptuelles”, Thèse de doctorat. Université d’Aix Marseille III, France – 1997.

(Furtado, 1999) FURTADO, M. E. S., “Integrando fatores humanos no processo de desenvolvimento de interfaces homem-computador adaptativas”, II Workshop sobre Fatores Humanos em Sistemas Computacionais, Campinas, SP, Brasil, 1999.

(Furtado et al., 2001) FURTADO, E., FURTADO, J. J. V., SILVA, W. B., RODRIGUES, D. W. T., TADDEO, L. da S., LIMBOURG, Q., VANDERDONCKT, J., “An ontology-based method for universal design of user interfaces”, Proceedings of Workshop on Multiple User Interfaces over the Internet: Engineering and Applications Trends, A. Seffah, T. Radhakrishnan & G. Canals (éds.), Lille, 10 Septembre 2001.

(Gama *et al.*, 2000) GAMA, E., HELM, R., JOHNSON, R., VLISSIDES, J., “Padrões de projeto: soluções reutilizáveis de software orientado a objetos”. Tradução Luiz A. Meirelles Salgado. Porto Alegre, Bookman, 2000

(Gamboa e Scapin, 1997) GAMBOA, F. R. and SCAPIN, D. L., “Editing MAD task descriptions for specifying user interfaces at both semantic and presentation levels”, in Proceedings DVS-IS’97, 4th International Eurographics Workshop on Design, Specification, and Verification of Interactive Systems, Granada – Spain, 1997.

(Gamboa, 1998) GAMBOA, F. R., “Spécification et implémentation d'ALACIE: Atelier Logiciel d'Aide à la Conception d'Interfaces Ergonomiques”, Thèse de Doctorat, Paris XI, Octobre, 1998.

(Guerrero e Lula, 2001) GUERRERO, C. V. S., LULA, B. Jr., “Um tutorial na Web: obtenção do modelo da interação a partir do modelo da tarefa com o auxílio de regras ergonômicas”, Relatório Técnico RT- DSC-002/2001 (55 páginas), UFPB/DSC/CCT Campina Grande, PB, outubro de 2001.

(Guerrero e Lula, 2002) GUERRERO, C. V. S., LULA, B. Jr., “Model-guided and task-based approach to UI design centered in a unified interaction and architectural model”, CADUI'2002 - 4th International Conference on Computer-Aided Design of User Interfaces, p. 107 - 119, Valenciennes, FRANCE, May 2002.

(Guerrero, 2002a) GUERRERO, C. V. S., "MEDITE - Uma metodologia orientada a modelos para a concepção de interfaces ergonômicas", Dissertação de Mestrado - COPIN, UFPB, Campina Grande PB, Fevereiro de 2002.

(Guerrero, 2002b) GUERRERO, D. D. S., "Redes de Petri Orientadas a Objeto". Tese de Doutorado, Pós-Graduação em Engenharia Elétrica, Universidade Federal da Paraíba (UFPB) – Campus II, Campina Grande, PB 2002.

(Hammouche, 1995) HAMMOUCHE, H., "De la modélisation des tâches utilisateurs au prototype de l'interface homme-machine", Thèse de Docteur, Université Paris VI, France, Décembre, 1995.

(Harel, 1987) HAREL, D. "Statecharts: a visual formalism for complex systems" in *Science of Computer Programming 8* (1987), 231-274.

(Harel, 1988) HAREL, D. "On visual formalisms". Communications of the ACM 1988; 31:514-530

(Hix e Hartson, 1988) HIX, D. , HARTSON, H., "Developing User Interfaces: Ensuring Usability Through Product & Process", John Wiley & Sons Inc, 1988.

(JAVA, 2003) Disponível em http://java.sun.com/ e acessado em [30/09/2003].

(JESS, 2003) Disponível em http://herzberg.ca.sandia.gov/jess/ e acessado em [30/09/2003].

(Jensen, 1992) JENSEN, K. "Coulored Petri Nets – Basic Concepts, Analysis Methods and Pratical Use – Vol 1.Ed. Spring-Verlag, USA, 1992

(Jensen, 1995) JENSEN, K. "Coulored Petri Nets – Basic Concepts, Analysis Methods and Pratical Use – Vol 2.Ed. Spring-Verlag, Germany, 1995

(Johnson e Johnson, 1989) JOHNSON, P., JOHNSON, H., "Knowledge Analysis of Task: Task Analysis and Specification for Human-Computer Systems" in Downton, A. (ed.): Enginnering the Human-Computer Interface. McGraw-Hill, Maidenhead (1989) 119-144.

(Johnson et al., 1993) JOHNSON, P., WILSON, S., MARKOPOULOS, P., PYCOCK, J., ADEPT – Advanced Design Environment for Prototyping with Task models, INTERCHI'93 Conference Proceedings, Amsterdam: ACM, 1993.

(Johnson et al., 1995) JOHNSON, P., JOHNSON, H., WILSON, S., "Rapid Prototyping of User Interfaces Driven by Task Models", in *Scenario-Based Design*: *Envisioning Work an Technology in System Development,* John Carroll (ed.), John Wiley & Sons, pp. 209-246, 1995

(Kafure, 2000) KAFURE, I. M., "Validação do Formalismo TAOS para a Concepção de Interfaces Homem-Computador", Dissertação de Mestrado - COPIN, Universidade Federal da Paraíba, Campina Grande, PB, 2000.

(Krumel, 2001a) KRUMEL, A., "JATO: A new kid on the open source block, Part 1 – A new library for converting between Java and XML", Java World (http://www.javaworld.com/javaworld/jw-03-2001/jw-0316-jato.html), March 2001. Acessado 01/03/2005

(Krumel, 2001b) KRUMEL, A., "JATO: A new kid on the open source block, Part 2 – Look in-depth at Java-to-XML translation", Java World (http://www.javaworld.com/javaworld/jw-04-2001/jw-0413-jato2.html), April 2001. Acessado 01/03/2005

(Krumel, 2001c) KRUMEL, A., "JATO: A new kid on the open source block, Part 3 – Translate XML documents into Java Objects", Java World (http://www.javaworld.com/javaworld/jw-05-2001/jw-0525-jato3.html), May 2001. Acessado 01/03/2005

(LFA, 2004) Liguagens Formais e Autômatos Home Page, Laboratório de Fundamentos da Computação, UFRGS, Porto Alegre, 2003, http://teia.inf.ufrgs.br/lformais, Acessado em 30/11/2004

(Limbourg et al., 2001) LIMBOURG, Q., PRIBEANU, C., VANDERDONCKT, J., "*Towards Uniformised Task Models in a Model Based Approach"*, in Proc. of 8th International Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'2001 (Glasgow, 13-15 juin 2001), Ch. Johnson (eds.), Lecture Notes in Computer Science, Vol. 2220, Springer-Verlag, Berlin, 2001, pp. 164-182.

(Lula, 1992) LULA, B. Jr., "Elaboration d'un Environnement de Génération Interactive d'Interfaces à Manipulation Directer pour le Language OBJLOG", Thèse de Docteur, Universidade de Droit d'Economie et des Sciences d'Aix-Marseille III, Faculté des Sciences et Techniques de Saint– Jérôme, França, 1992.

(Markopoulos et al., 1992) MARKOPOULOS, P., PYCOCK, J., WILSON, S., "ADEPT - A

task based design environment", Queen Mary and Westfield College, UK, 1992.

(Medeiros, 1995) MEDEIROS H., "*L'Utilisation d'un Language Terminologique dans la Modélisation de Concepts Dynamiques"*; Journées 1995, Projet ACNOS, LRPS-BETA-LAG-INRIA, 04-05, Sept 1995, Strasbourg, França, 1995.

(Medeiros e Rousselot, 1995) MEDEIROS, H. e ROUSSELOT F., "Un Outil D'Aide à la Modélisation de Concepts Dynamiques: Le Système TAME; Journées Acquisition - Validation –Apprentissage", JAVA'95, 04/95, Grenoble, França, 1995.

(Medeiros e Rousselot, 1995a) MEDEIROS, H. e ROUSSELOT, F., "*Acquisition et Modélistation de Concepts Dynamiques: Le Système TAME"*, Rapport ERIC R0102-96, Strasbourg, França, 1995.

(Medeiros e Rousselot, 1995b) MEDEIROS, H. e ROUSSELOT F., "*Un Outil D'Aide à la Modélisation de Concepts Dynamiques: Le Système TAME"*; Journées Acquisition - Validation - Apprentissage, JAVA'95, 04/95, Grenoble, França, 1995.

(Medeiros *et al.*, 2000) MEDEIROS, H., KAFURE, I. M e LULA, B. Jr*.,* "TAOS: a Task- and Action Oriented Framework for User's Task Analysis in the Context of Human-Computer Interfaces", Proceeding of SCCC 2000 – XX International Conference oh he Chilean Computer Science Society, november 2000, Santiago, Chile, 2000.

(Medeiros et al., 2002[a]) MEDEIROS, F. P. A., CORDEIRO, P. B. e LULA, B. J., "Projeto iTAOS - Modelagem da tarefa e Fase de planejamento". Relatório Técnico RT- DSC-001/2002, (112 p.), UFPB/DSC/CCT Campina Grande, PB, Brazil, abril, 2002, disponível em www.dsc.ufcg.edu.br/~itaos.

(Medeiros et al., 2002b) MEDEIROS, F. P. A., CORDEIRO, P. B. e LULA, B. J., "Projeto iTAOS - Fase de Implementação (Primeira Iteração) e Especificação Conceitual da Interação". Relatório Técnico RT-DSC-003/2002, (47 p.), UFPB/DSC/CCT Campina Grande, PB, Brazil, julho, 2002, disponível em www.dsc.ufcg.edu.br/~itaos.

(Medeiros et al., 2002c) MEDEIROS, F. P. A., LULA, B. e CORDEIRO, P. B. A "Graphical Tool to Support Task Description using TAOS Formalism for UI Design". In: 7th ERCIM Workshop, 2002, Paris. 7th ERCIM Workshop. ERCIM (European Research Consortium for Informatics and Mathematics), 2002. p.45 - 51.

(Medeiros et al., 2002d) MEDEIROS, F. P. A.; LULA, B., CORDEIRO; P. B., "iTAOS: a Graphical Tool to Support User's Task Description in UI Design Context". In: V Symposium on Human Factors in Computer Systems, 2002, Fortaleza.V Symposium on Human Factors in Computer Systems. Fortaleza: BNDE, 2002. v.01. p.376 - 379.

(Medeiros, 2003) MEDEIROS, F. P. A., "Projeto e implementação do módulo TAOS-Graph da ferramenta iTAOS para análise e modelagem da tarefa", Dissertação de Mestrado - COPIN, UFCG, Campina Grande PB, Fevereiro de 2003.

(Murata, 1989) MURATA, T. "Petri Nets: Properties, Analysis and Applications". Proceedings of the IEEE, no 77 (4), p.541-580, Abril de 1989

(Norman, 1983) NORMAN, D. A., "Some Observations on Mental Models. Mental Models", 7-14. Editado por Dedre Gentner & Albert L. Stevens, Lawrence Erlbaum Associates, Hillsdale, New Jersey. 1983.

(Norman, 1986) NORMAN, D. A., "Cognitive Engineering", User Centered Systems Design (Ed. Norman, D. A. et Draper, S.) Lawrence Erlbaum Associates Ltd., 1986.

(Pfaff, 1985) PFAFF, G. R. "User Interface Management Systems", Eurographics Seminars, Springer-Verlag, 224 pages, 1985.

(Papert, 1980) PAPERT, S., "Mindstorms: Children, Computers, and Powerful Ideas", Basic Books, Inck, New York, 1980.

(Paterno *et al.*, 1997) PATERNÒ, F., MANCINI, C., MENICONI, S., "ConcurTask-Trees: A Diagrammatic Notation for Specifying Task Models", *Proc. Of INTERACT'97,* pp. 362-369. 1997

(Rodrigues et al., 2003b) RODRIGUES, C. E. C. L., SUÁREZ, P. R., JÚNIOR, B. L., OLIVEIRA, R. C. L. de, "Obtenção do Modelo da Interação a partir do Modelo de Cenário", Relatório Técnico DSC-006/03. Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Departamento de Sistemas e Computação. COPIN, 2003.

(Rodrigues e Lula, 2004) RODRIGUES, C. E. C. L., LULA, B. J., "Introduzindo o modelo de roteiro na metodologia MEDITE e estendendo o modelo de interação EDITOR" em Anais do III WDCOPIN - III Workshop de Dissertações da COPIN, Campina Grande, Paraíba, Brasil, 16 -17 de agosto de 2004, Paginas 19 – 24.

(Rumbaugh et al., 1998) RUMBAUGH, J., JACOBSON, I., BOOCH, G. "The unified modeling language reference manual", Addison Wesley Longman , Massachusetts, 1998

(Scapin, 1987) Scapin D. L., "Guide Ergonomique de Conception des Interfaces Homme-Machine", Rapport INRIA No. 77, France, 1987.

(Scapin e Pierret-Golbreich, 1989) SCAPIN, D., PIERRET-GOLBREICH, C., "Towards a method for task description: MAD" in Berlinguet, L., Berthelette, D. (eds.): Proc. Of Conf. Work with Display Units WWU'89, Elsevier Science Publishers, Amsterdam (1989) 27-34.

(Silva, 2000) SILVA, P. P. da, "User Interface Declarative Models and Development Environments: A Survey". In *Interactive Systems: Design, Specification, and Verification* (7th International Workshop DSV-IS, Limerick, Ireland, June, 2000), Ph. Palanque and F. Paternò (Eds.). LNCS Vol. 1946, pages 207-226, Springer, 2000.

(Smith *et al.*, 1987) SMITH R. G., BARTH, P. S. et YOUNG, R. L., "A substrate for Object-Oriented Interface Design", Research Direction in Object-Oriented Programming (Ed. Shriver&Wegner), Computer Systems Serie, MIT Press, 1987.

(Sousa, 1999) SOUSA, M. R. F. de, "Avaliação iterativa da especificação de interfaces com ênfase na navegação", Tese de Doutorado, Coordenação de Pós-Graduação em Engenharia Elétrica - COPELE, UFPB, Campina Grande, dezembro de 1999.

(Suárez et al., 2003a) SUÁREZ, P. R., JÚNIOR, B. L., BARROS, M. A., "Uma Estratégia de Gestão do Conhecimento para o Projeto de Interfaces Homem-Computador", KMBRASIL'03, São Paulo, 2003.

(Suárez et al., 2003b) SUÁREZ, P. R., JÚNIOR, B. L., RODRIGUES, C. E. C. L., GOMES, H. M., BARROS, M. A. de, SANTOS, S. M., "Representação do Conhecimento e Inteligência Artificial: Uma Estratégia de Apoio ao Projeto de Interfaces Homem-Computador", KMBRASIL'03, São Paulo, 2003.

(Suárez et al., 2003c). SUÁREZ, P. R., JÚNIOR, B. L., BARROS, M. A., "Meta-Modelos de Tarefa, Usuário, Cenário e Interação para o Projeto de IHC", Relatório Técnico DSC-005/03. Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Departamento de Sistemas e Computação. COPIN, 2003.

(Suárez, 2004) SUÁREZ, P. R., "Gestão do Conhecimento no Processo de Concepção de IHC e uma Nova Abordagem para a Obtenção de

uma Especificação Conceitual da Interação". Dissertação de Mestrado - COPIN, UFCG, Campina Grande PB, Fevereiro de 2004.

(Suárez *et. al.*, 2004) SUÁREZ, P. R., LULA, B. J., BARROS, M. A., "Applying Knowledge Management in UI Design Process" in Proceedings of the 3rd annual conference on Task models and diagrams 2004, Prague, Czech Republic *November 15 - 16,* Pages: 113 – 120, 2004.

(Tse, 1991) TSE T., PONG, L. "An examination of requirements specification languages". Computer Journal 1991;34:143-152

(UML v1.5, 2004) disponível em www.omg.org/technology/documents/formal/uml.htm acessado em [10/09/2004]

(Vanderdonckt e Bodart, 1994) VANDERDONCKT, J., BODART, F., "Jusqu'au bout avec nos règles ergonomiques", in Actes des Sixièmes Journées sur l'Ingénierie des Interfaces Homme-machine IHM'94 (Lille, 8-9 December 1994), pp. 231-236.

(Vanderdonckt *et al.*, 2002) VANDERDONCKT, J., MBAKI, E.,"Window Transitions: A Graphical Notation for Specifying Mid-level Dialogue", in Proceedings of 1st International Workshop on Task Models and Diagrams for user interface design TAMODIA'2002 (Bucarest, 18-19 juillet 2002), Academy of Economic Studies of Bucharest, INFOREC Printing House, Bucharest, 2002, pp. 55-63.

(Vanderdonckt *et al.*, 2003) VANDERDONCKT, J., LIMBOURG Q., FLORINS, M., "Deriving the navigational structure of a user Interface". In *Proceedings of the 9th IFIP TC 13 Int.Conference on Human-Computer Interaction Interact2003 Zürich 1–5 September 200*3, pages 455–462, 2003.

(Vasconcelos, 2004) VASCONCELOS, C. R., "XPU – Um Modelo Para o Desenvolvimento de Sistemas Centrado no Usuário" Dissertação de Mestrado - COPIN, UFCG, Campina Grande PB, Fevereiro de 2004.

(van Welie, 2001) van Welie. M, "Task-based User Interface Design", Universiteit Amsterdam, Ph.D. Thesis, 2001

(Ward e Mellor, 1985) WARD, P.; MELLOR, S., "Structured Development for Real-time Systems", 1985, Prentice Hall, pp. 86, 302.

(Wilson et al., 1993) WILSON, S., JOHNSON, P., KELLY, C., CUNNHINGAM, J.,

MARKOPOULOUS, P., “Beyond hacking: a model based approach to user interface design”, In Proceedings of HCI'93, J. Alty, D. Diaper and S. Guest (eds), Cambridge University Press, 1993.

(XMI, 2004) www.omg.org/technology/documents/formal/xmi.htm Acessado em [30/11/2004]

(XML, 2005) www.w3.org/XML/ Acessado em [02/02/2005]

ANEXO A

Este anexo apresenta detalhes do diálogo intra-EDITOR. Como dito no capítulo 4 o diálogo intra-EDITOR é obtido através dos atributos *método* das tarefas associadas a cada um dos elementos da árvore EDITOR. Os operadores SEQ, OR, XOR, AND, SIM e PAR, são os possíveis valores assumidos pelo atributo *método* e representam quais as seqüências, restrições e condições de realização de uma tarefa, implicando nas mesmas condições de realização (transição) dos estados do statechart. A combinação entre os atributos *ocorrência* (que nos fornece os eventos possíveis a cada um dos elementos do modelo EDITOR) e *método* (seqüência, restrições e condições das transições) nos dá todas as transações internas possíveis ao super estado de um statechart, ou seja, transações intra-EDITOR. O Quadro (a.1) representa todas as combinações *método* x *ocorrência* possíveis.

Quadro (a.1): Combinações possíveis *ocorrência* versus *método*

		Método de Realização das Sub-tarefas.					
		SEQ	**XOR**	**OR**	**AND**	**SIM**	**PAR**
Ocorrência	**(0,1)**	X	X	X	-	X	X
	(0,n)	X	X	X	-	X	X
	(1,1)	X	X	X	X	X	X
	(1,n)	X	X	X	X	X	X

É importante salientar que a maior parte das transações é oriunda de eventos dos objetos de interação. As visões e espaços, também lançam eventos, entretanto, esses eventos representam uma não realização ou repetição dos estados que representam estes elementos (visões e espaços). Dessa forma, vamos exemplificar as condições e restrições de cada operador *método* de realização das tarefas através dos eventos dos objetos de interação.

A árvore EDITOR levada em consideração nesse anexo é representada na figura (a.1). Os statecharts apresentados nessa seção não apresentam o super estado que representa o elemento EDITOR, apenas os estados correspondesntes a visão (*visão* A) e aos objetos de interação (*O.I.* B, *O.I.* C, *O.I.* D) são representados.

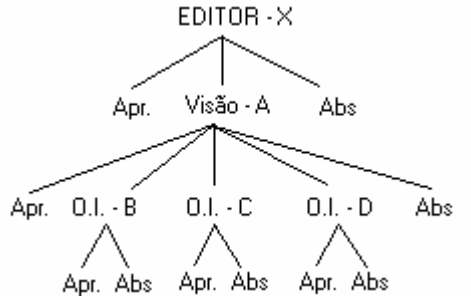


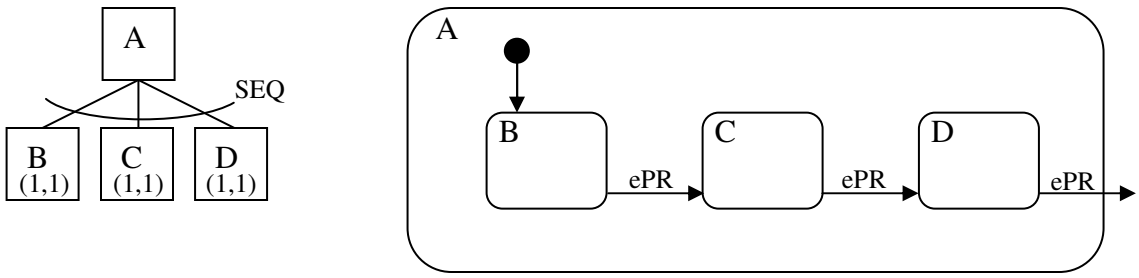
Figura (a.1): Árvore Editor

1.1.1. *Método* SEQ

O *método* SEQ determina que todas as sub-tarefas, de uma tarefa pai, devem ser realizadas em sequência, sempre da esquerda para a direita. Por exemplo, se uma Tarefa A, possui 3 sub-tarefas B, C, D regidas pelo operador SEQ, significa dizer será realizada primeiramente B, seguida por C e finalmente por D. Caso algumas dessas sub-tarefas contenham sub-tarefas, antes de passar para a próxima, suas filhas tem que ser realizadas (de acordo com o o atributo *método* respectivo).

- *Ocorrência* (1,1)

Com a ocorrência (1,1) a tarefa tem que ser realizada apenas uma vez, portanto, o evento gerado é obrigatoriamente ePR (evento Procedimento Realizado) como esta exemplicidado na figura abaixo

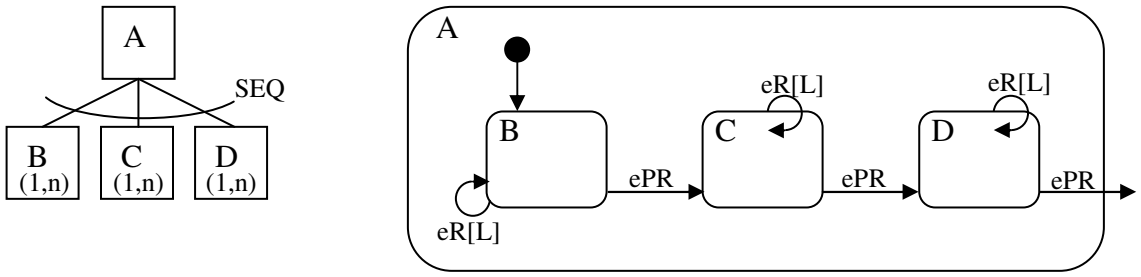


ePR - evento Procedimento(s) Realizado(s)

- *Ocorrência* (1,n)

Com a ocorrência (1,n) a tarefa tem que ser realizada pelo menos uma vez e no máximo n vezes, portanto, os eventos possíveis são, ePR (evento Procedimento Realizado)

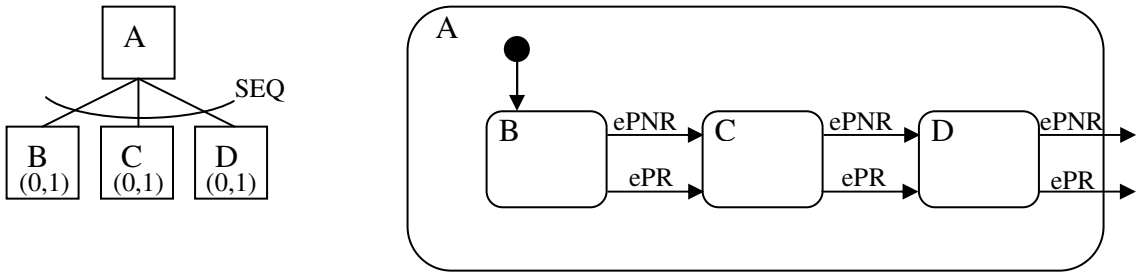
e eR (evento Repetir). eR conta com um guarda [L] que verifica se o número máximo de repetições já foi alcançado. A figura abaixo ilustra a situação.



ePR - evento Procedimento(s) Realizado(s)
eR - eventoRepetir
[L] = estáDentroDoLimite()

- *Ocorrência* (0,1)

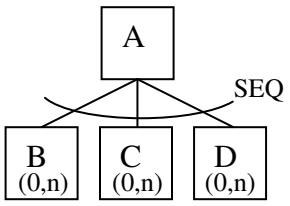
Com a ocorrência (0,1) a tarefa pode não ser realizada, ou seja, é uma tarefa opcional ou então só pode ser realizada apenas uma única vez. Os eventos possíveis portanto são ePNR (evento Procedimento não Realizado) ou ePR (evento Procedimento Realizado). A figura abaixo ilustra o caso.

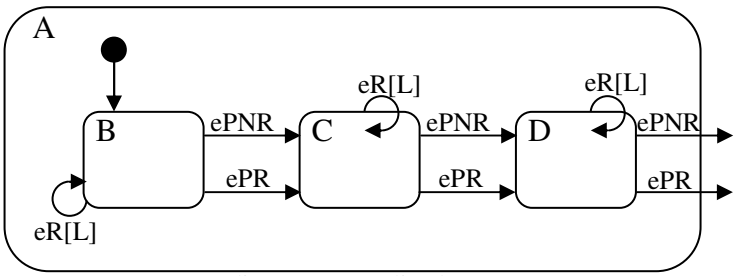


ePR - evento Procedimento(s) Realizado(s)
ePNR - evento Procedimento(s) não Realizado(s)
ePNR / AnulaAçõesEAtividadesDaOrigem();

- *Ocorrência* (0,n)

Com a ocorrência (0,n) a tarefa pode não ser realizada, ou seja, é uma tarefa opcional ou então pode ser realizada n vezes. Os eventos possíveis portanto são ePNR (evento Procedimento não Realizado), ePR (evento Procedimento Realizado) ou eR (evento Repetir). A figura abaixo ilustra o caso.





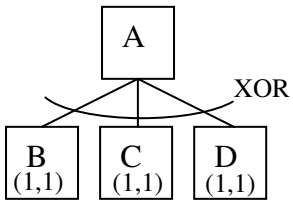
ePR - evento Procedimento(s) Realizado(s)
ePNR - evento Procedimento(s) não Realizado(s)
ePNR / AnulaAçõesEAtividadesDaOrigem();
eR - eventoRepetir
[L] = estáDentroDoLimite()

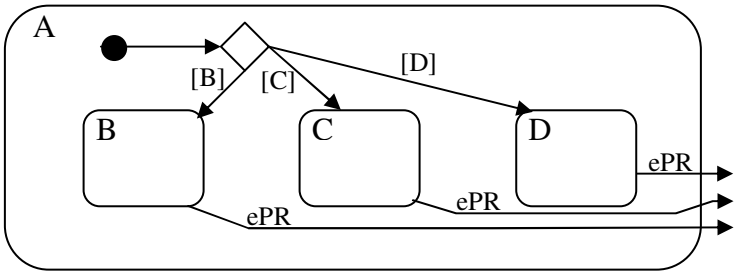
1.1.2. *Método* XOR

O método XOR preconiza que apenas uma das sub-tarefas de uma tarefa pai deve ser realizada.

- *Ocorrência* (1,1)

Com a ocorrência (1,1) a tarefa tem que ser realizada apenas uma vez, portanto, o evento gerado é obrigatoriamente ePR (evento Procedimento Realizado). A figura abaixo demonstra os elementos que irão compor um pedaço do statechat. Notem que o pseudo-estado inicial leva a um pseudo-estado condicional que encaminhará o objeto a um único sub-estado – por meio da avaliação dos guardas –, assim que a tarefa associada a cada um desses sub-estados for realizada, o evento ePR é gerado, havendo, então, uma transição para um outro estado destino.

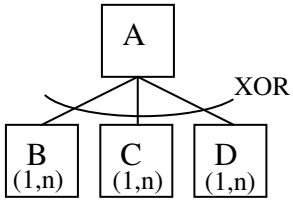


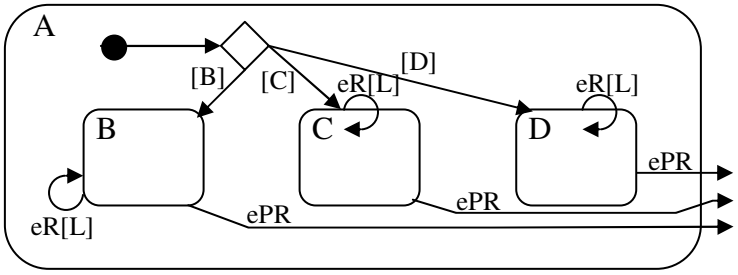


ePR - evento Procedimento(s) Realizado(s) [B] = escolhido_B();
[C] = escolhido_C();
[D] = escolhido_D();

- *Ocorrência* (1,n)

Com a ocorrência (1,n) a tarefa tem que ser realizada pelo menos uma vez e no máximo n vezes, portanto, os eventos possíveis são, ePR (evento Procedimento Realizado) e eR (evento Repetir). eR conta com um guarda [L] que verifica se o número máximo de repetições já foi alcançado. A figura abaixo representa esta situação.

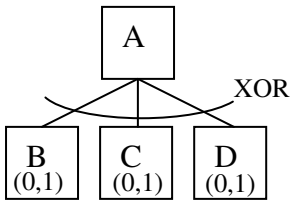


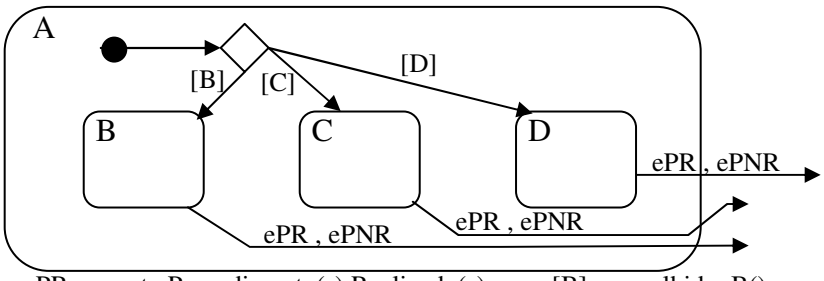


ePR - evento Procedimento(s) Realizado(s) [B] = escolhido_B();
eR - eventoRepetir [C] = escolhido_C();
[L] = estáDentroDoLimite() [D] = escolhido_D();

- *Ocorrência* (0,1)

A combinação dos operadores XOR e (0,1) significa dizer que, apenas uma tarefa pode ser realizada uma única vez ou então, nenhuma tarefa será realizada. Os eventos, neste caso, são ePR – caso a tarefa escolhida seja executada uma vez – e o evento ePNR – caso a tarefa escolhida não seja executada. A figura abaixo ilustra a situação



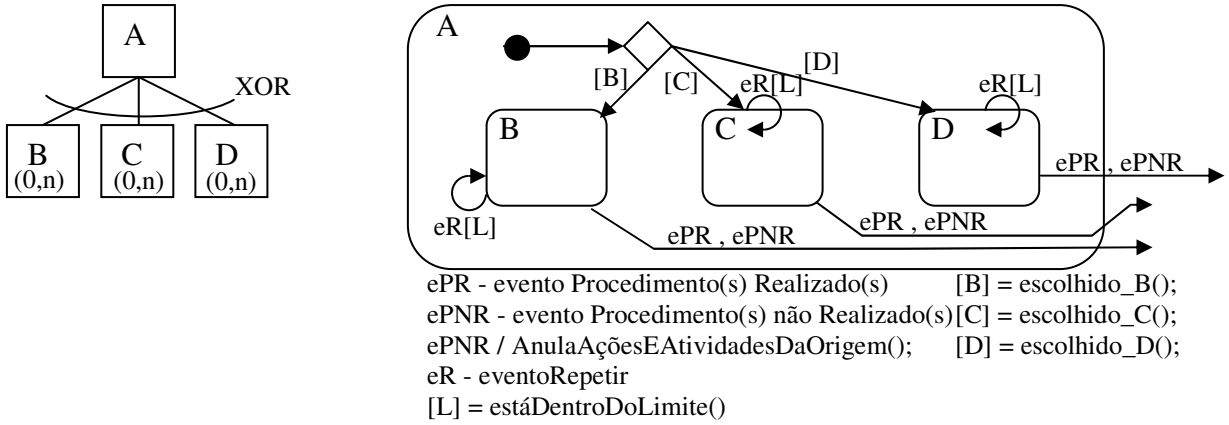


ePR - evento Procedimento(s) Realizado(s) [B] = escolhido_B();
ePNR - evento Procedimento(s) não Realizado(s) [C] = escolhido_C();
ePNR / AnulaAçõesEAtividadesDaOrigem(); [D] = escolhido_D();

- *Ocorrência* (0,n)

A combinação dos operadores XOR e (0,n) significa dizer que, a tarefa pode não ser realizada ou ser realizada até n vezes. Os eventos, neste caso, são ePR – caso a tarefa

escolhida seja executada uma vez – , eR[L] – caso o usuário deseje repetir até n vezes a tarefa –, e o evento ePNR – caso a tarefa escolhida não seja executada. A figura abaixo ilustra a situação



1.1.3. *Método* OR

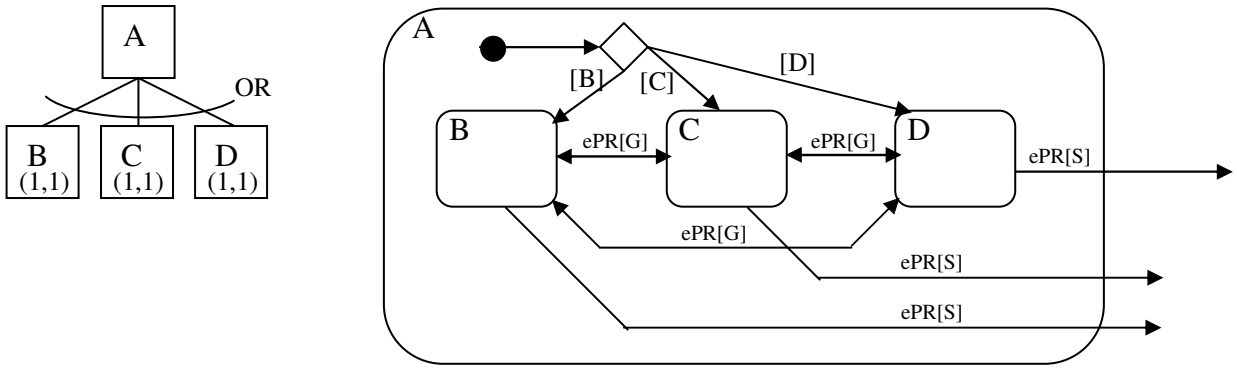
Utilizar o *método* OR significa dizer que nenhuma, algumas, ou todas as tarefas podem ser realizadas. Neste caso, não importa a ordem que elas devam ser realizadas.

- *Ocorrência* (1,1)

A combinação do *método* OR com a *ocorrência* (1,1) é semelhante à utilização da combinação *método* AND com *ocorrência* (1,1). Utilizar esta combinação significa dizer que a tarefa tem que ser executada, obrigatoriamente, uma vez[1]. A figura abaixo ilustra as transações possíveis para o caso. O único evento possível é o ePR, entretanto, esse evento possui diferentes guardas, possibilitando transições pertinentes a este caso. O guarda [G] avalia duas funções lógicas ("ehEstadoDestinoEscolhido()" e "estadoDestinoNãoUltrapassouLimite()"), a primeira verifica se determinado estado foi escolhido como de destino. A segunda cláusula verifica se o estado de destino não ultrapassou o limite de vezes em que ele poderia estar ativo (ou seja, se a tarefa ainda não ultrapassou a o limite de sua ocorrência). O guarda [S] também avalia duas funções lógicas("ehEstadoDestinoEscolhido()" e "condiçõesDeOcorrênciaDasSubTarefasSatisteitas()"). A

[1] Um exemplo dessa situação seria o preenchimento de um formulário eletrônico, onde a tarefa pai (preencher formulário) seria realizada com o *método* OR de "preencher nome", "preencher endereço" e "preencher telefone". Se qualquer uma das sub-tarefas possuir *ocorrência* (1,1) significa dizer que ela é uma tarefa obrigatória.

primeira já foi explicada e a segunda é um controle para saber se todos os estados que tenham associadas tarefas com *ocorrência* (1,1) ou (1,n) já foram visitados (já estiveram ativos). Esse controle é necessário para que não se saia do *superestado* OR sem que a condição de realização obrigatória da tarefa seja satisfeita. A transição, portanto, irá ocorrer ao estado que satisfizer a algum dos guardas. Caso nenhum guarda seja satisfeito, o evento é descartado e o estado de origem continua ativo.

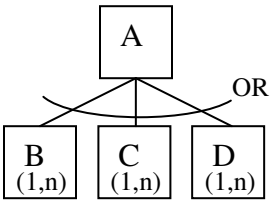


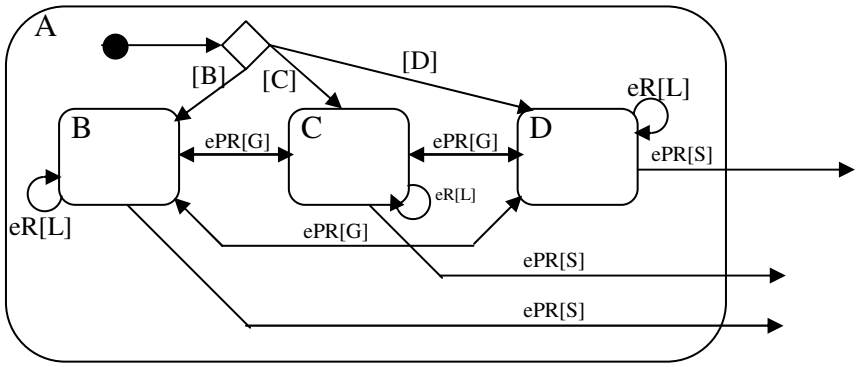
[G] = ehEstadoDestinoEscolhido() e estadoDestinoNãoUltrapassouLimite();
[S] = ehEstadoDestinoEscolhido() e condiçõesDeOcorrênciaDasSubTarefasSatisteitas();
ePR - evento Procedimento(s) Realizado(s)

[B] = escolhido_B();
[C] = escolhido_C();
[D] = escolhido_D();

- *Ocorrência* (1,n)

A combinação *método* XOR com *ocorrência* (1,n) é quase igual a combinação anterior (XOR, (1,1)). A diferença está no fato que cada um pode ser visitado até n vezes. O único evento a mais, é o evento eR (possibilitando até n visitas os sub-estados). A figura abaixo ilustra essa situação.

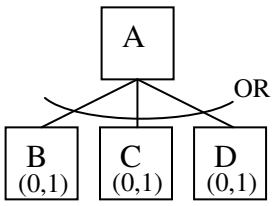


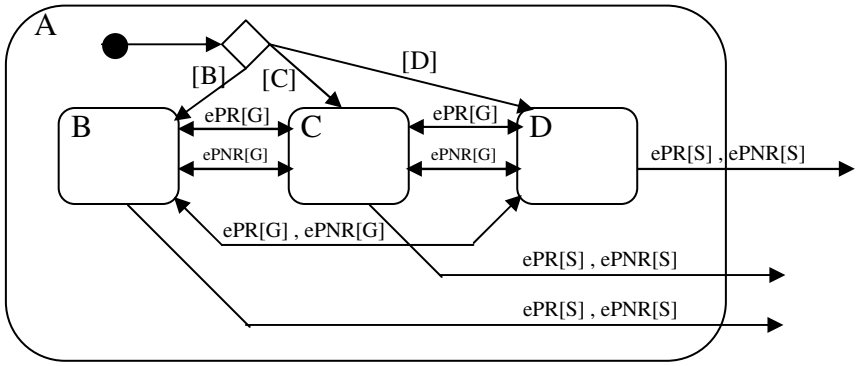


[G] = ehEstadoDestinoEscolhido() e estadoDestinoNãoUltrapassouLimite();
[S] = ehEstadoDestinoEscolhido() e condiçõesDeOcorrênciaDasSubTarefasSatisteitas();
ePR - evento Procedimento(s) Realizado(s) [B] = escolhido_B();
eR - eventoRepetir [C] = escolhido_C();
[L] = estáDentroDoLimite(); [D] = escolhido_D();

- *Ocorrência* (0,1)

Nesta situação, a tarefa pode não ser realizada ou então ser realizada apenas uma vez. Dois eventos são previstos neste caso. ePR e ePNR, acompanhados com os guardas [G] ou [S]. Se o estado estiver ativo, mas a tarefa associada ao estado não for realizada, o evento ePNR será disparado. Caso contrário (tarefa realizada) o evento ePR será lançado. A análise dos guardas que determinará qual transição ocorrerá.

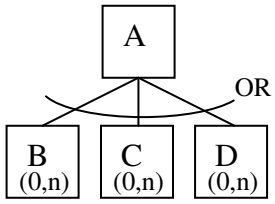


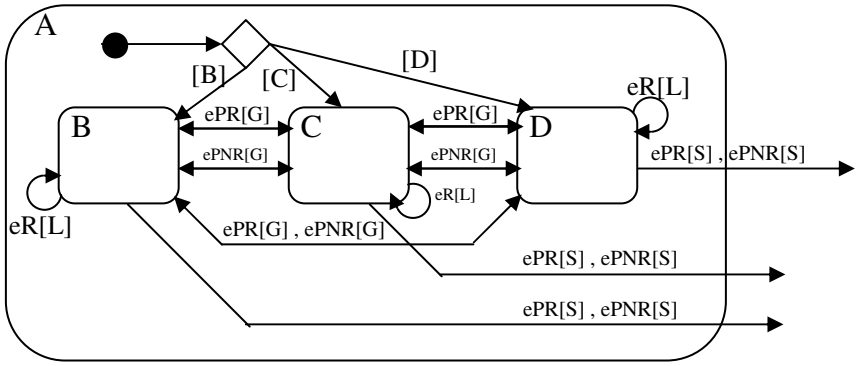


[G] = ehEstadoDestinoEscolhido() e estadoDestinoNãoUltrapassouLimite();
[S] = ehEstadoDestinoEscolhido() e condiçõesDeOcorrênciaDasSubTarefasSatisteitas();
ePR - evento Procedimento(s) Realizado(s) [B] = escolhido_B();
ePNR - evento Procedimento(s) não Realizado(s) [C] = escolhido_C();
ePNR / AnulaAçõesEAtividadesDaOrigem(); [D] = escolhido_D();

- *Ocorrência* (0,n)

Semelhante a situação anterior, difere apenas que o estado pode ser visitado até n vezes. Inclui-se neste caso o evento eR com guarda [L].





[G] = ehEstadoDestinoEscolhido() e estadoDestinoNãoUltrapassouLimite();
[S] = ehEstadoDestinoEscolhido() e condiçõesDeOcorrênciaDasSubTarefasSatisteitas();
ePR - evento Procedimento(s) Realizado(s) [B] = escolhido_B();
ePNR - evento Procedimento(s) não Realizado(s) [C] = escolhido_C();
ePNR / AnulaAçõesEAtividadesDaOrigem(); [D] = escolhido_D();
eR - eventoRepetir
[L] = estáDentroDoLimite();

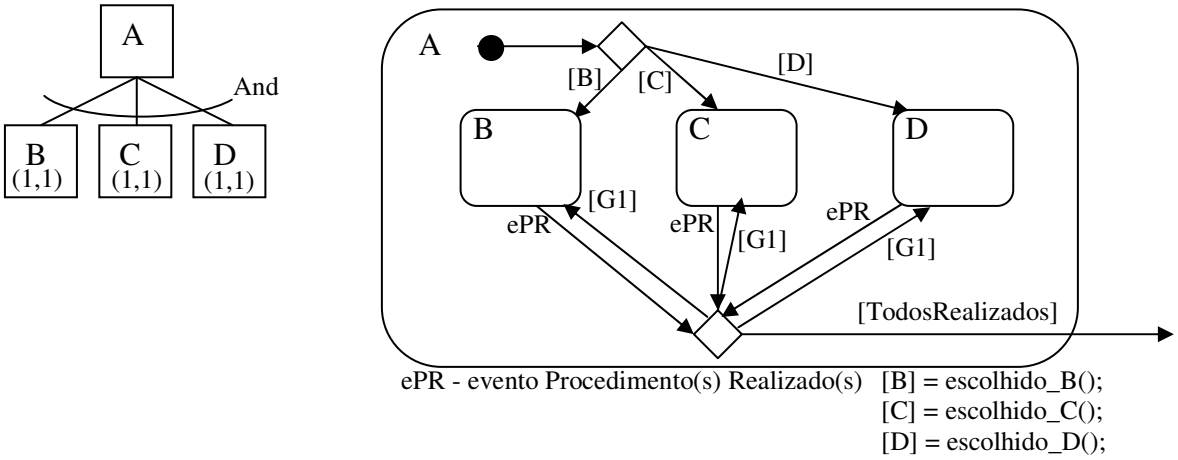
1.1.4. *Método* AND

O método AND implica dizer que todas as sub-tarefas que por ele forem regidas tem que ser realizadas obrigatoriamente. Por esse motivo, combinações {AND x (0,1)} e {AND x (0,n)} não são possíveis de serem realizadas.

- *Ocorrência* (1,1)

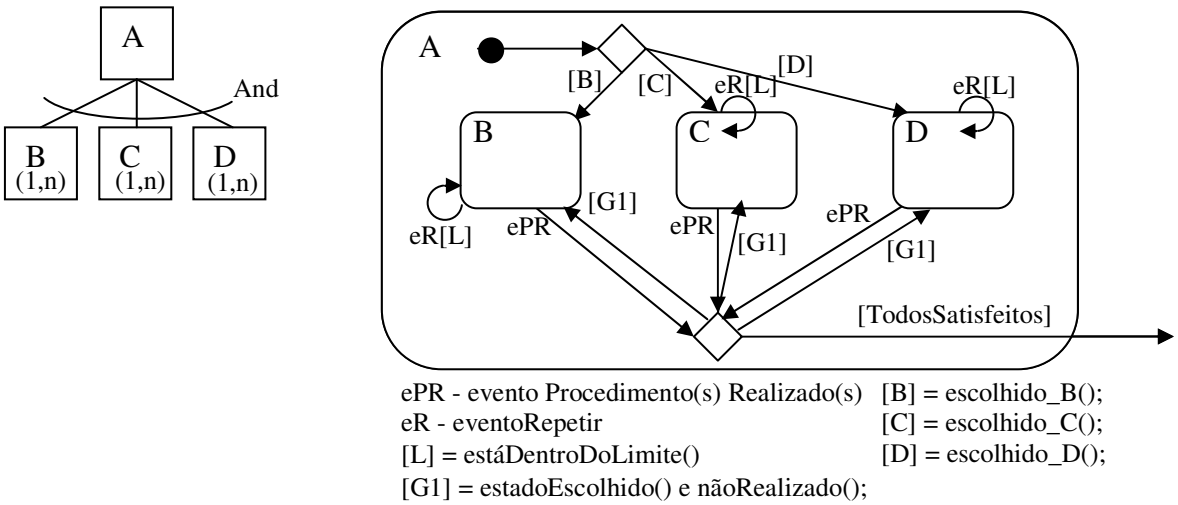
A combinação {AND x (1,1)} informa que cada estado tem que ser visitado e apenas uma única vez, independentemente da ordem realização. Como todos os estados têm que ser visitados, apenas um evento neste caso é lançado – ePR. Quando o evento ePR é lançado o objeto vai ao pseudo-estado condicional para ver qual será o estado de destino. A avaliação é feita através do guarda [G1], que possui duas cláusulas (“estadoEscolhido()” e “estadoNãoRealizado();”). A primeira cláusula avalia qual o estado que está sendo escolhido e a segunda avalia se o estado desejado ainda não foi realizado, ou seja, se o

estado ainda não tiver sido visitado, o mesmo pode ser visitado. Quando todos os elementos estiverem realizados, o superestado é deixado. A figura abaixo ilustra essa situação.



- *Ocorrência* (1,n)

A ocorrência (1,n) com o método AND é quase igual a anterior. A diferença é que um estado pode ser realizado mais de uma vez (antes de ser deixado). Portanto, o evento eRL indica essa possibilidade enquanto o guarda [L] for verdadeiro (tal guarda verifica o se o número máximo de ocorrências não foi ultrapassado). Quando todos os estados forem satisfeitos, o superestado é deixado.

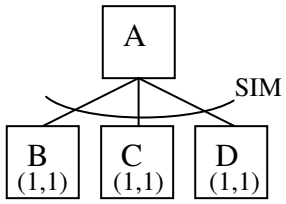


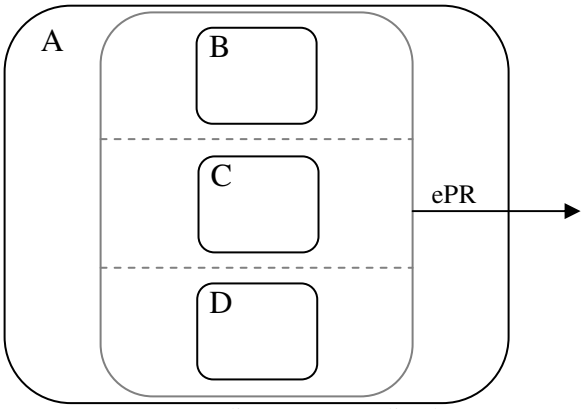
1.1.5. *Método* SIM

Sub-tarefas regidas pelo *método* SIM representam uma das maneiras de concorrência entre tarefas. O operador de simultaneidade – SIM – requer que todas as tarefas comecem em um mesmo instante e o que os finais das tarefas também coincidam.

- *Ocorrência* (1,1)

Neste caso, B, C e D começam ao mesmo tempo e terminam ao mesmo tempo. Essas tarefas são realizadas uma única vez. Portanto apenas o evento ePR será disparado quando todas elas terminarem.

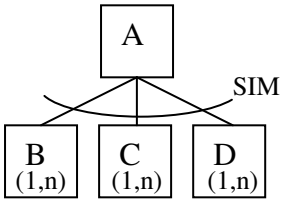


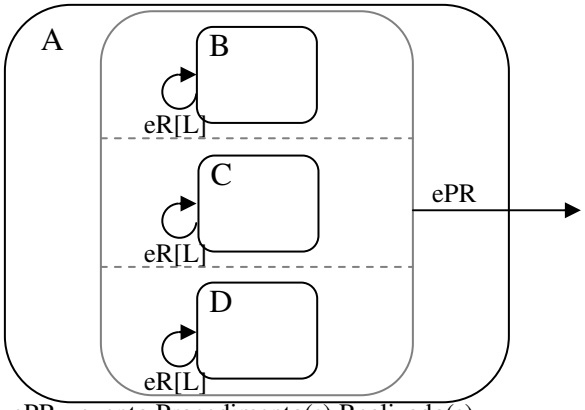


ePR - evento Procedimento(s) Realizado(s)

- *Ocorrência* (1,n)

Neste caso, por serem tarefas simultâneas (requerem que começos coincidentes e finais coincidentes), ou todas são realizadas uma vez, ou todas são realizadas n vezes. Os eventos ePR e eR são previstos nesta configuração.

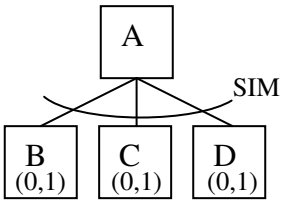


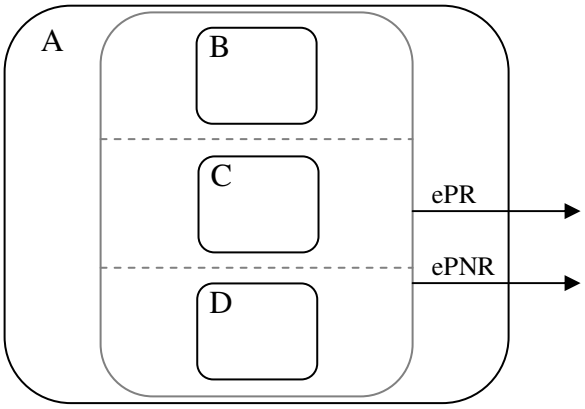


ePR - evento Procedimento(s) Realizado(s)
eR - eventoRepetir
[L] = estáDentroDoLimite()

- *Ocorrência* (0,1)

A possibilidade de "zero" realizações indica que as tarefas são opcionais. Se todas forem realizadas "0" (zero) vezes, implica dizer que o evento Procedimento(s) não Realizado(s) será disparado. Observando a figura abaixo, se B e C, por exemplo, forem realizadas "0" vezes, e D "1" (uma) vez, ao final da realização de D, o evento disparado será ePR, pois as tarefas B e C foram tidas como opcionais. Portanto, no exemplo dado, após o lançamento de ePR o objeto sairá do estado composto ortogonal para um outro estado de destino.

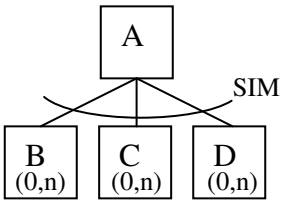


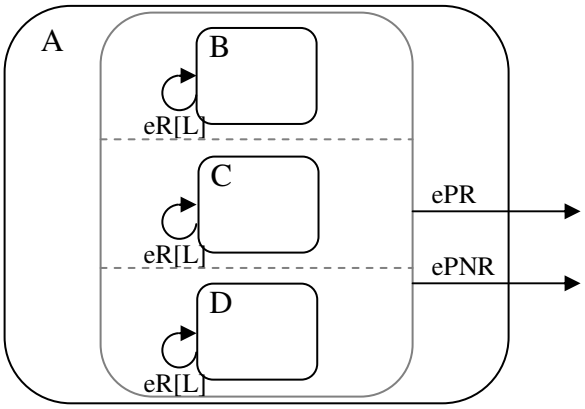


ePR - evento Procedimento(s) Realizado(s)
ePNR - evento Procedimento(s) não Realizado(s)
ePNR / AnulaAçõesEAtividadesDaOrigem();

- *Ocorrência* (0,n)

Se todas forem realizadas "0" (zero) vezes, implica dizer que o evento Procedimento(s) não Realizado(s) será disparado (ePNR). As regiões ortogonais têm que ser realizadas "0" ou "n" vezes, Se por exemplo, B for realizado "0" vezes, e C e D n vezes, o Estado Ortogonal só será deixado depois que C e D forem realizadas as "n" vezes. Note que, por serem simultâneas, as "n" vezes de C, tem que ser iguais as "n" vezes de D.





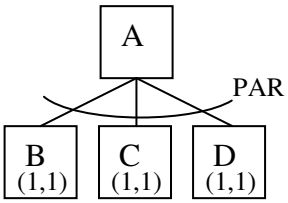
ePR - evento Procedimento(s) Realizado(s)
ePNR - evento Procedimento(s) não Realizado(s)
ePNR / AnulaAçõesEAtividadesDaOrigem();
eR - eventoRepetir
[L] = estáDentroDoLimite()

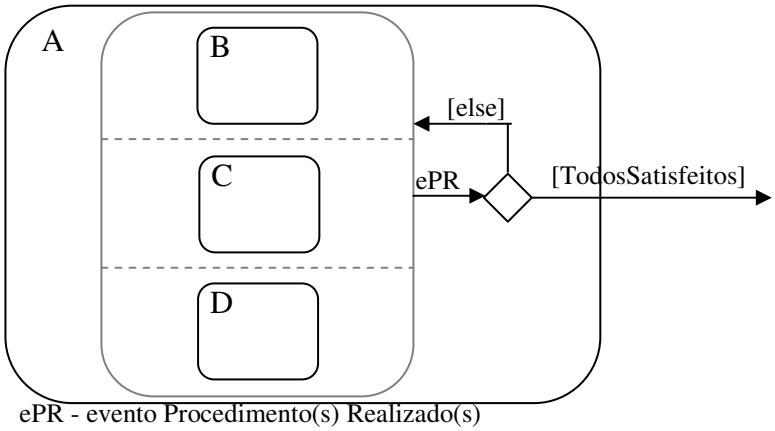
1.1.6. *Método* PAR

O operador de Paralelismo (PAR) indica que as tarefas são realizadas concorrentemente, sendo que, o final das tarefas não precisa coincidir. Entretanto, por se tratar de tarefas concorrentes, todas elas precisam estar satisfeitas para que haja a transição do estado ortogonal a outro estado de destino.

- Ocorrência (1,1)

Toda vez que uma região ortogonal (do estado composto ortogonal) é satisfeita, um evento ePR é gerado . Se o guarda [TodosSatisfeitos] for verdadeiro, o estado composto é deixado para outro de estado de destino. Se alguma região ortogonal já foi satisfeita (realizada uma vez, neste caso), mas ainda há outras regiões ortogonais sendo realizadas (ainda não foram satisfeitas) o objeto continua no Estado Ortogonal, embora não mais na região ortogonal já satisfeita.

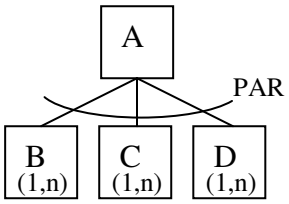


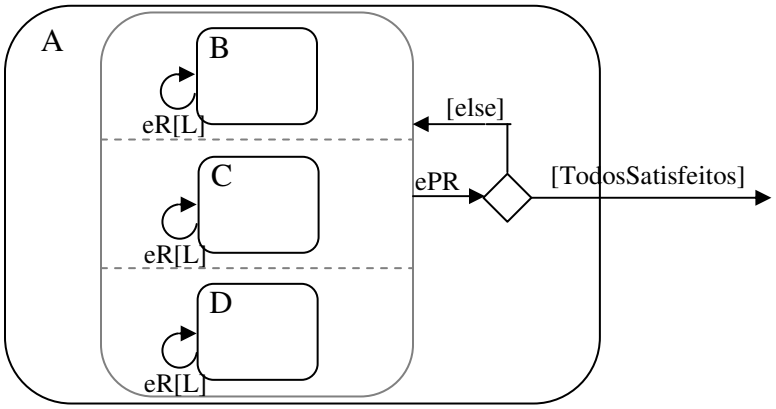


ePR - evento Procedimento(s) Realizado(s)

- Ocorrência (1,n)

Todas as regiões ortogonais precisam estar satisfeitas para que haja a transição do estado ortogonal a outro estado de destino. Toda vez que uma região ortogonal (do estado composto ortogonal) é satisfeita, um evento ePR é gerado . Se o guarda [TodosSatisfeitos] for verdadeiro, o estado composto é deixado para outro de estado de destino. Aqui, todos precisam ser realizados "1" ou "n" vezes.



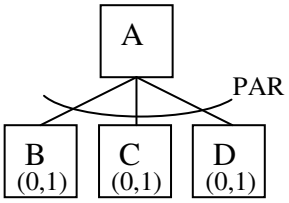


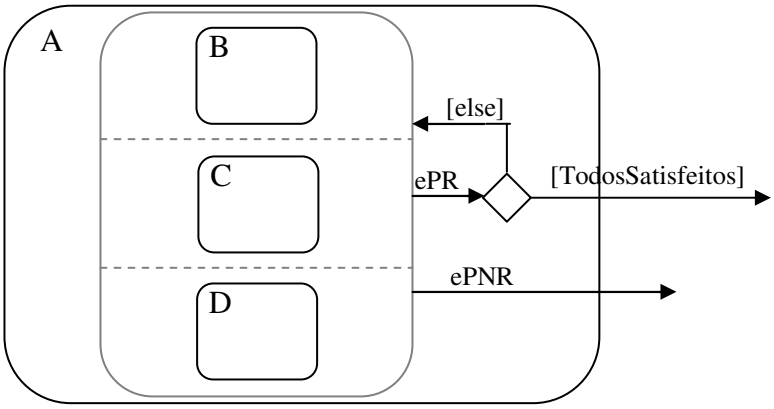
ePR - evento Procedimento(s) Realizado(s)
eR - eventoRepetir
[L] = estáDentroDoLimite()

- *Ocorrência* (0,1)

Toda vez que uma região ortogonal (do estado composto ortogonal) é satisfeita, um evento ePR é gerado . Se o guarda [TodosSatisfeitos] for verdadeiro, o estado composto é deixado para outro de estado de destino. Se alguma região ortogonal for realizada "0" (zero) vezes, então seu status será de satisfeita (neste caso, pelo menos uma das outras

regiões ortogonais têm que ser realizadas “1” vez). Se todas forem realizadas “0” (zero) vezes, então o evento ePNR será disparado.

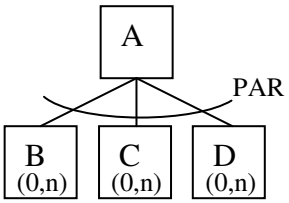


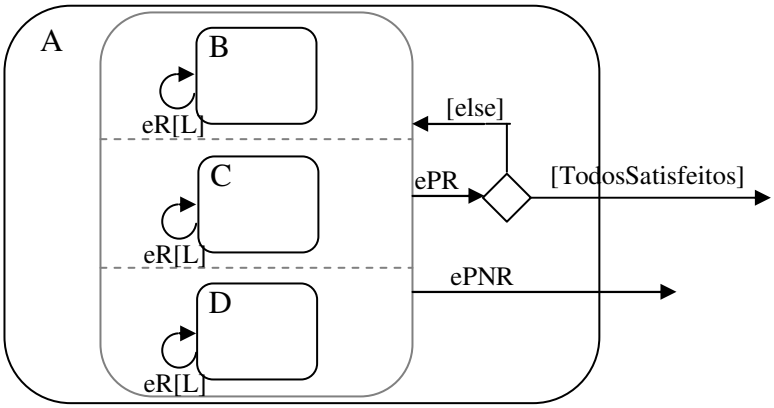


ePR - evento Procedimento(s) Realizado(s)
ePNR - evento Procedimento(s) não Realizado(s)
ePNR / AnulaAçõesEAtividadesDaOrigem();

- *Ocorrência* (0,n)

Toda vez que uma região ortogonal (do estado composto ortogonal) é satisfeita, um evento ePR é gerado . Se o guarda [TodosSatisfeitos] for verdadeiro, o estado composto é deixado para outro de estado de destino. Se todas forem realizadas “0” (zero) vezes, então o evento ePNR será disparado. Uma Região ortogonal só será considerada satisfeita, depois que ela for realizada n vezes.





ePR - evento Procedimento(s) Realizado(s)
ePNR - evento Procedimento(s) não Realizado(s)
ePNR / AnulaAçõesEAtividadesDaOrigem();
eR - eventoRepetir
[L] = estáDentroDoLimite()

ANEXO B

Os algoritmos básicos de manipulação de elementos do modelo da interação e que serão objeto de propagação de volta ao modelo da tarefa (mecanismos de manutenção da coerência) são as seguintes:

1. Inclusão
   - 1.1. Incluir *Objeto de interação* (*Espaço de Interação*)
   - 1.2. Incluir *Visão (Espaço de Interação)*
   - 1.3. Incluir *Espaço de Interação*
     - 1.3.1. filho do *Espaço Inicial*
     - 1.3.2. filho de *Espaço de Interação*
     - 1.3.3. filho de *Espaço de Direcionamento*
   - 1.4. Incluir *Espaço de Direcionamento* com rearranjo (de outros *Espaços*)
     - 1.4.1. filho do *Espaço Inicial*
     - 1.4.2. filho do *Espaço de Direcionamento*
2. Exclusão
   - 2.1. Excluir *Objeto de interação* (se houver link, excluir também o *espaço* associado).
   - 2.2. Excluir *Visão*
   - 2.3. Excluir *Espaço de Interação*
     - 2.3.1. Filho do *Espaço Inicial*
     - 2.3.2. Filho de *Espaço de Direcionamento*
     - 2.3.3. Filho de *Espaço de Interação*
   - 2.4. Excluir *Espaço de Direcionamento* (sem excluir seus filhos)
     - 2.4.1. Excluir *Espaço de Direcionamento* Re-arranjando filhos no *Espaço Inicial*
     - 2.4.2. Excluir *Espaço de Direcionamento* Re-arranjando filhos no *Espaço de Direcionamento*

Abaixo, serão demonstrados todos os mecanismos de manutenção da coerência entre os modelos com uma descrição, o algoritmo e um exemplo da pré e pós-aplicação de cada mecanismo.

1.1 - Incluir *Objeto de interação* (*Espaço de Interação*)

Descrição: Este algoritmo inclui um *Objeto de Interação* em um *Espaço de Interação*. O *Objeto de Interação* em questão é para realizar um procedimento, como uma confirmação, a aquisição de uma informação (inserir um dado, por exemplo), etc, ou seja, este objeto não é para realizar simplesmente um link para outro *Espaço* do *Modelo da Interação*.

Algoritmo: **Incluir** - *Objeto de Interação* {Objeto de interação realiza um procedimento}
MAPEAMENTO INTERAÇÃO -> ROTEIRO

1. Identificar o *Espaço E (de Interação)* onde o objeto será inserido *(Mint.).*
   1.1. Identificar o *Cenário C(MRo.)* associado ao *E(Mint.).*
2. Identificar a *Visão Vx* de *E* onde o objeto será inserido*(Mint.).*
   2.1. Identificar a *Cena Cnx (MRo.)* associada à *Vx (Mint.).*
3. Criar um *Objeto de Interação Oix*
4. SE a *Visão Vx(Mint.)* tiver um *Objeto de Interação default* (vazio) *Oid (Mint.)*
   ENTÃO Substituir Oid por *Oix* na *Visão Vx* de *E (Mint.).*
   SENÃO Inserir o *Objeto de Interação Oix* na *Visão Vx* de *E (Mint.).*
5. Criar uma *Tomada Tox (MRo.).*
   5.1. Inserir *Tox(MRo.)* como filha de *Cnx(MRo.).*
6. Associar o *Objeto de Interação Oix (Mint.)* à *Tomada Tox (MRo.).*
7. Definir a ocorrência e o Método de Realização de *Oix (Mint.)*

MAPEAMENTO ROTEIRO-> TAREFA

8. SE A *Cena CNx* (*MRo*.) tiver apenas uma *Tomada Tox* (*MRo.*) e *Tox* estiver associada a uma *Tarefa* (antes de inserir a nova tomada)
   ENTÃO

8.1. Lançar exceção informando que a *visão Vx* associada à *Cena Cnx* não pode receber mais um objeto de interação pois já abriga um objeto de interação que faz link a outro Espaço*.

9. SE A Cena CNx (MRo.) tiver apenas uma Tomada Tox (MRo.) (antes de inserir a nova tomada)

   ENTÃO

   9.1. Identificar a *Ação Ax (MTa.)* associada à *Cena Cnx (MRo.)*

   9.2. Identificar a *Tarefa TM (MTa)* mãe da *Ação Ax (MTa.)*

   9.3. Inserir em *TM* uma *Tarefa Tx (MTa.)*

   9.4. Associar *Tx (MTa.)* à *Cnx (MRo.)* e Desassociar Ax *(MTa.)* de Cnx *(MRo.)*

   9.5. Retirar *Ax (MTa.)* de TM *(MTa.)* e inserir-la em TX *(MTa.)*

   9.6. Criar uma *Ação Af (MTa.)* como filha de *Tx (MTa.)*

   9.7. Associar *Tox (MRo.)* à *Af (MTa)*.

   9.8. Modificar o atributo *Método* da *Tarefa Tx (MTa.)* inserindo *Ax e Af (MTa.).*.

   9.9. Extrair o diálogo *de Tx (MTa.)* através do atributo *Método* e *Ocorrência.*

10. SE a *Cena Cnx (MRo.)* tiver mais de uma *Tomada Tox (MRo.)* (antes de inserir a nova tomada)

    ENTÃO

    10.1. Identificar a *Tarefa Tx (MTa.)* associada à *Cnx (MRo.)*

    10.2. Criar e Associar uma *Ação Ax (MTa.)* para a *Tomada Tox (MRo).*

    10.2.1. *Criar e Inserir Pré-Situação de Ax (MTa.).*

    10.2.2. *Criar e Inserir Pós-Situação de Ax (MTa.).*

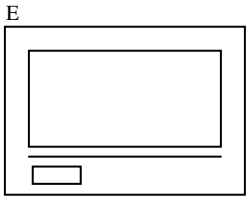
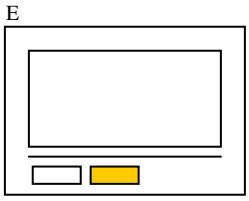
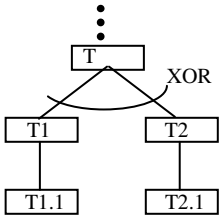
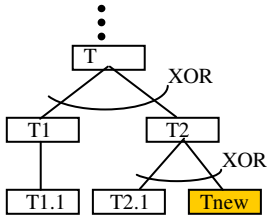
    10.3. Adicionar a *Ação Ax (MTa.)* como filha da *Tarefa Tx (MTa.).*

    10.3.1. Modificar o atributo *Método* da *Tarefa Tx (MTa.)* inserindo *Ax (MTa.).*.

    10.3.2. Extrair o diálogo *de Tx (MTa.)* através do atributo *Método* e *Ocorrência.*

*Há uma exeção neste caso. Se a visão possuir um objeto de interação que faça link a outro espaço, então ela não pode conter mais nenhum objeto de interação.

Exemplo:

Antes	Depois
Interação	*Interação*
	
Tarefa	*Tarefa*
	

1.2 - Incluir *Visão (Espaço de Interação)*

Descrição: Este algoritmo inclui uma *Visão* para conter *Objetos de Interação* em um *Espaço de Interação*. Os *Objetos de Interação* em questão é para realizar um procedimento, como uma confirmação, a aquisição de uma informação (inserir um dado, por exemplo), etc, ou seja, este objeto não é para realizar simplesmente um link para outro *Espaço* do *Modelo da Interação*.

Algoritmo: Incluir - **Visão** *(*em *Espaço de Interação)*

MAPEAMENTO INTERAÇÃO -> CENÁRIO

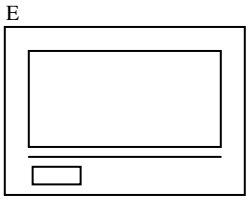
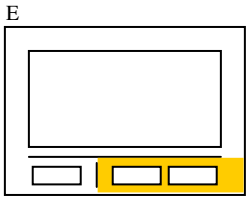
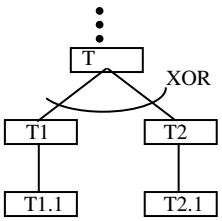
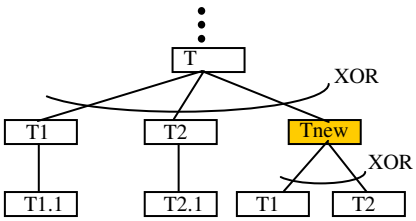
1. Identificar o *Espaço E (de Interação)* no *(MInt.)*.
   1.1. Identificar o *Cenário C (MRo.)* associado ao *Espaço E (Mint.).*
2. Inserir uma *Visão Vx (MInt.)* no *Espaço E (MInt.).*

3. Criar uma *Cena Cnx (MRo.) como filha do Cenario C (MRo.).*
4. Associar *Vx (Mint) à Cnx (MRo.).*
5. Definir a ocorrência e o Método de Realização de *Vx(Mint.)*

MAPEAMENTO CENÁRIO -> TAREFA

6. Identificar a *Tarefa TM (MTa.)* associado ao *Cenário C (MRo.)*
7. Criar uma *Tarefa Tx (MTa.)* no modelo de Tarefa
   7.1. Criar pré e pós-condições de *Tx (MTa.)*
8. Adicionar a *Tarefa Tx (MTa.)* como filha da *Tarefa TM (MTa.)*
   8.1. Modificar o atributo Método da *Tarefa TM (MTa.)* inserindo *Tx (MTa.)*
   8.2. Extrair o diálogo de *TM (MTa.)* através do atributo Método e Ocorrência.
9. Associar a *Cena CNx (MRo.)* à *Tarefa Tx (MTa.)*
10. Adicionar o *Objeto de Interação Oix (Mint.)* em *Vx (Mint.)* – **Regra inserir OI *ou um Objeto de Interação default (Vazio) Oid.***

Exemplo:

Antes	**Depois**
Interação	*Interação*
	
Tarefa	*Tarefa*
	

1.3 Incluir *Espaço de Interação*

Os algoritmos desta seção dizem respeito à inserção de um *Espaço de Interação* no *Modelo de Interação*. Um *Espaço de Interação* pode ser inserido a partir do *Espaço Inicial* ("filho" do *Espaço Inicial*) a partir de um *Espaço de Direcionamento* ("filho" de um *Espaço de Direcionamento*) ou a partir de outro *Espaço de Interação* ("filho" de um *Espaço de Interação*). Abaixo as definições de cada um destes espaços (estas também estão no capítulo 5):

Espaço de Interação: Pelo menos um de seus elementos (*Objeto de Interação* e/ou *Visão*) é associado a uma tarefa elementar da aplicação. É formado por *Visões* (associadas a *tarefas elementares* ou *não elementares*) e *Objetos de Interação* (associadas a *tarefas elementares* ou *não elementares*).

***Espaço de Direcionamento*:** O *Espaço de Direcionamento* é um *Espaço* cuja tarefa a ele associada é pai de tarefas que estejam associadas a outros *Espaços* (todas as suas sub-tarefas estão associadas a *Espaços*). Ou seja, é um *Espaço* "pai" de outro(s) *Espaço*(s). Como o nome sugere, ele serve apenas para prover o direcionamento (a navegação) do de um *Espaço* até outros *Espaço*s *de Interação* (serão definidos adiante). Dependendo da profundidade da árvore da tarefa, poderá haver um *Espaço* de *Direcionamento* que seja "pai" de outro *Espaço de Direcionamento*. Pode-se afirmar, também, que os *Objetos de Interação* desse tipo de *Espaço* não estão associados a tarefas elementares.

Espaço Inicial: Representa a tela inicia da aplicação. Contém duas visões:

1. *Visão Funcionalidade* – Contém *objetos de interação* que representarão os *links* a *Espaços de Interação* ou *de Direcionamento*.

2. *Visão Orientação* – Esta *Visão* contém um *Objeto de Interação* para exposição de algum conteúdo (inicial) ao usuário da aplicação, como, por exemplo, uma mensagem (pode ser um texto, uma figura, texto com figura, etc). A *Visão Orientação* será sobreposta pelos outros *Espaço*s da aplicação, quando eles forem acionados através da *Visão Funcionalidade*, criando assim o conceito de *Área de Trabalho*.

Para inserir um *Espaço de Interação* no *Modelo da Interação*, é necessário criá-lo, antes de tudo, portanto, deve-se chamar a função de **CriarEspaçoDeInteração** para que um *Espaço de Interação* seja criado, assim como toda sua estrutura de *Roteiro* e *Tarefa*.

CriarEspaçoDeInteração

Mapeamento *Elementos da INTERAÇÃO -> Elementos do ROTEIRO*

1. Criar um *Espaço de Interação E*
   1.1. Criar e Inserir n *Visões V1, V2,...,Vx* no *Espaço de Interação E*
   1.2. Criar e Inserir n *Objetos de Interação Oix* em cada *Visão Vx* de *E*.
2. Criar um *Cenário C* e associá-lo ao *Espaço de Interação E*.
   2.1. Para cada *Visão Vx* de *E* Criar e Inserir em no *Cenário C* uma *Cena Cnx*
      2.1.1. Associar cada *Visão Vx* a sua *Cena Cnx* correspondente
   2.2. Para cada *Objeto de Interação Oix* Criar e Inserir na *Cena Cnx* uma *Tomada Tox*
      2.2.1. Associar cada *Objeto de Interação Oix* a sua *Tomada Tox* correspondente

Mapeamento *Elementos do ROTEIRO -> Elementos da TAREFA*

3. Criar uma *Tarefa T* e associa-la ao *Cenário C*
   3.1. Para cada *Cena Cnx* de *C* Criar e Inserir uma *Tarefa Tfx* na *Tarefa T*
      3.1.1. Associar cada *Cena Cnx* a sua *Tarefa Tfx* correspondente
   3.2. Para cada *Tomada Tox* de *Cnx* Criar e Inserir uma *Ação Ax* na *Tarefa Tfx*
   3.3. Associar cada *Tomada Tox* a sua *Ação Ax* correspondente

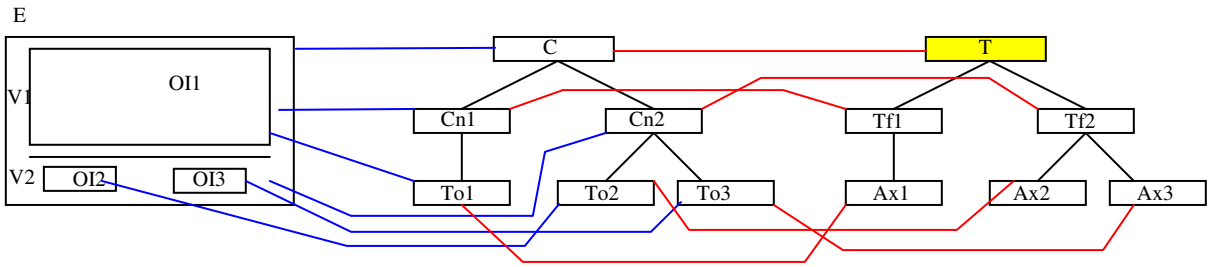


Ilustração do algoritmo de criação de *Espaço de Interação*

1.3.1 Incluir *Espaço de Interação filho do Espaço Inicial*

Descrição: Este algoritmo inclui um *Espaço de Interação* diretamente do *Espaço Inicial*. Para isso, mais um *Objeto de Interação* é criado na *Visão Funcionalidade* do *Espaço Inicial* para fazer link ao *Espaço Interação* que está sendo criado.

Algoritmo: Inserir um *Espaço de Interação* Filho do *Espaço Inicial*

1. CriarEspaçoDeInteração *Efilho*

Criação e Mapeamento de Elementos INTERAÇÃO -> ROTEIRO

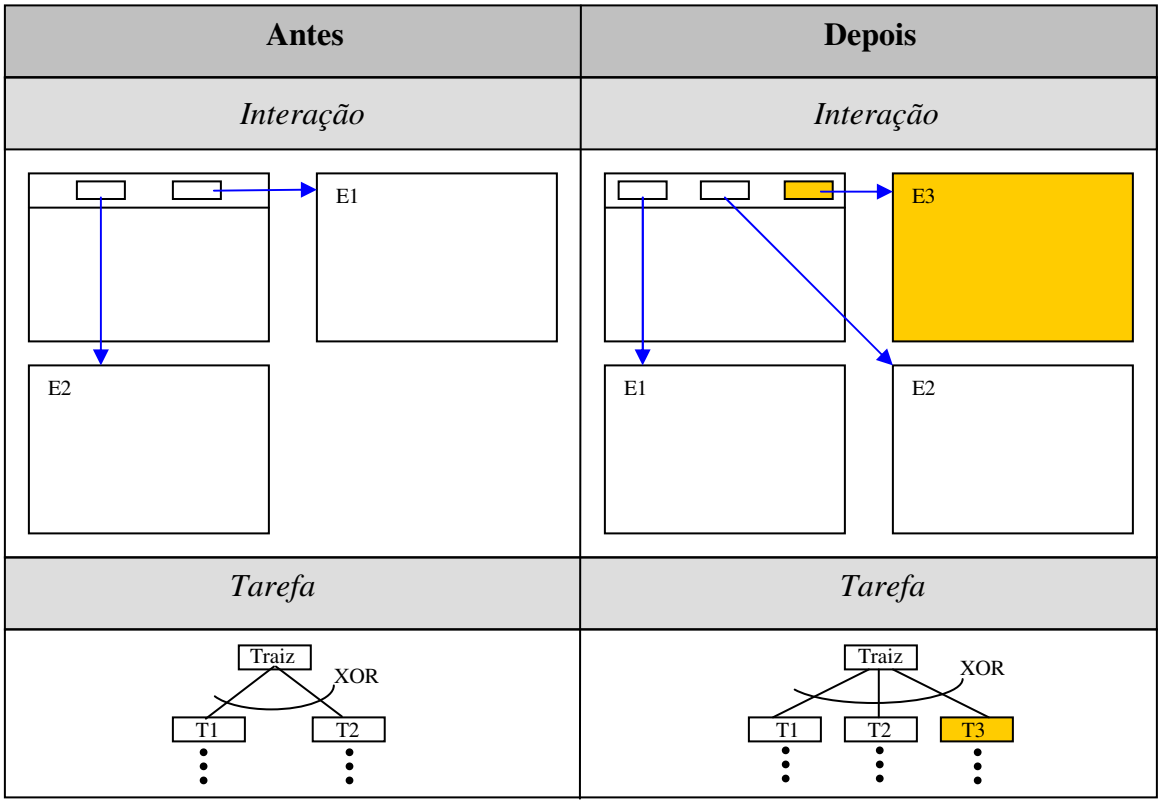
1. Introduzir o *Espaço de Interação Efilho* na árvore de Interação Abaixo do *Espaço Inicial Einicial*
2. Identificar o *Cenário Cinicial* associado ao *Einicial*
3. Identificar a *Visão Vfuncionalidade* do *Einicial*
4. Criar e Inserir um *Objeto de Interação OiLink* na *Vfuncionalidade*
5. Identificar a *Cena CnFuncionalidade* do *Cinicial*
6. Criar e Inserir uma *Tomada ToLink* na *Cinicial*
7. Associar *ToLink* ao *OiLink*

Criação e Mapeamento de Elementos ROTEIRO -> TAREFA

1. Identificar o *Cenário Cfilho* associado ao *Espaço de Interação Efilho*
2. Introduzir o *Cenário Cfilho* na árvore de Roteiro Abaixo do *Cenário Cinicial*
3. Identificar a *Tarefa Tfilha* associada ao *Cfilho*
4. Identificar a *Tarefa Traiz* associada ao *Cinicial*

5. Inserir a *Tarefa Tfilha* como uma Sub-Tarefa da *Tarefa Traiz* na posição indicada
6. Modificar o atributo *Método* da *Tarefa Traiz* inserindo a nova subtarefa *Tfilha* regida pelo mesmo operador de suas irmãs (XOR)
7. Associar o *Cenário Cfilho* a *Tarefa Tfilha*

Exemplo:



1.3.2 – Inserir *Espaço de Interação* filho de *Espaço de Interação*

Descrição: Este algoritmo inclui um *Espaço de Interação* diretamente de outro *Espaço de Interação*. Para isso, mais uma *Visão* e um *Objeto de Interação* são criados no do *Espaço de Interação de Origem* para fazer link ao *Espaço Interação de Destino* que está sendo criado.

Algoritmo: Inserir um *Espaço de Interação* Filho de um *Espaço de Interação*

1. CriarEspaçoDeInteração *Efilho*

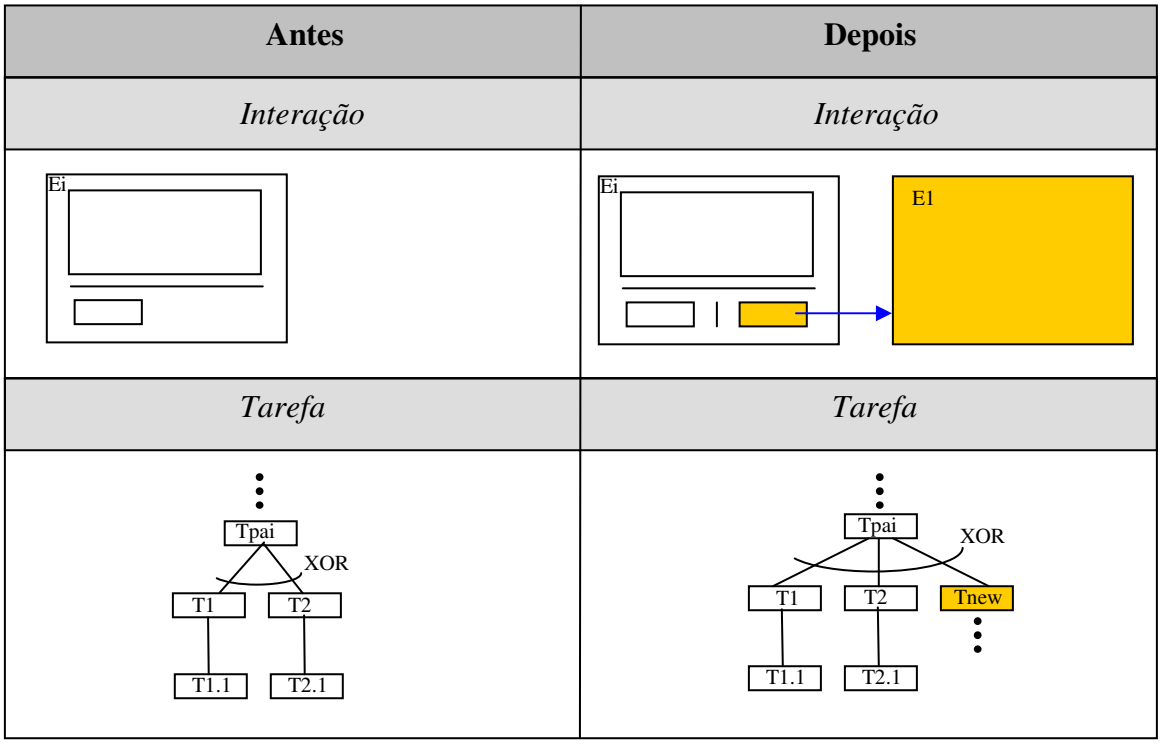
Criação e Mapeamento de Elementos INTERAÇÃO -> ROTEIRO

1. Introduzir o *Espaço de Interação Efilho* na árvore de Interação Abaixo do *Espaço de Interação Epai*
2. Identificar o *Cenário Cpai* associado ao *Espaço de Interação Epai*
3. Criar e Inserir uma *Visão Vlink* no *Epai*
4. Criar e Inserir um *Objeto de Interação OiLink* na *Vlink*
5. Criar e Inserir uma *Cena CnLink* no *Cpai*
6. Criar e Inserir uma *Tomada ToLink* na *CnLink*
7. Associar *CnLink* à *Vlink*
8. Associar *ToLink* ao *OiLink*

Criação e Mapeamento de Elementos ROTEIRO -> TAREFA

1. Identificar o *Cenário Cfilho* associado ao *Espaço de Interação Efilho*
2. Introduzir o *Cenário Cfilho* na árvore de Roteiro Abaixo do *Cenário Cpai*
3. Identificar a *Tarefa Tfilha* associada ao *Cfilho*
4. Identificar a *Tarefa Tpai* associada ao *Cpai*
5. Inserir a *Tarefa Tfilha* como uma Sub-Tarefa da *Tarefa Tpai* na posição indicada
6. Modificar o atributo *Método* da *Tarefa Tpai* inserindo a nova subtarefa *Tfilha* regida pelo mesmo operador de suas irmãs
7. Associar a *Cena CLink* a *Tarefa Tfilha*
8. Associar a *Tomada ToLink* a *Tarefa Tfilha*
9. Associar o *Cenário Cfilho* a *Tarefa Tfilha*

Exemplo:



1.3.3 – Inserir *Espaço de Interação* filho de *Espaço de Direcionamento*

Descrição: Este algoritmo inclui um *Espaço de Interação* diretamente de um *Espaço de Direcionamento*. Para isso, mais uma *Visão* e um *Objeto de Interação* são criados no do *Espaço de Direcionamento* (de Origem) para fazer link ao *Espaço Interação* (de Destino) que está sendo criado.

Algoritmo: Inserir um *Espaço de Interação* Filho de um *Espaço de Direcionamento*

1. CriarEspaçoDeInteração *Efilho*

Criação e Mapeamento de Elementos INTERAÇÃO -> ROTEIRO

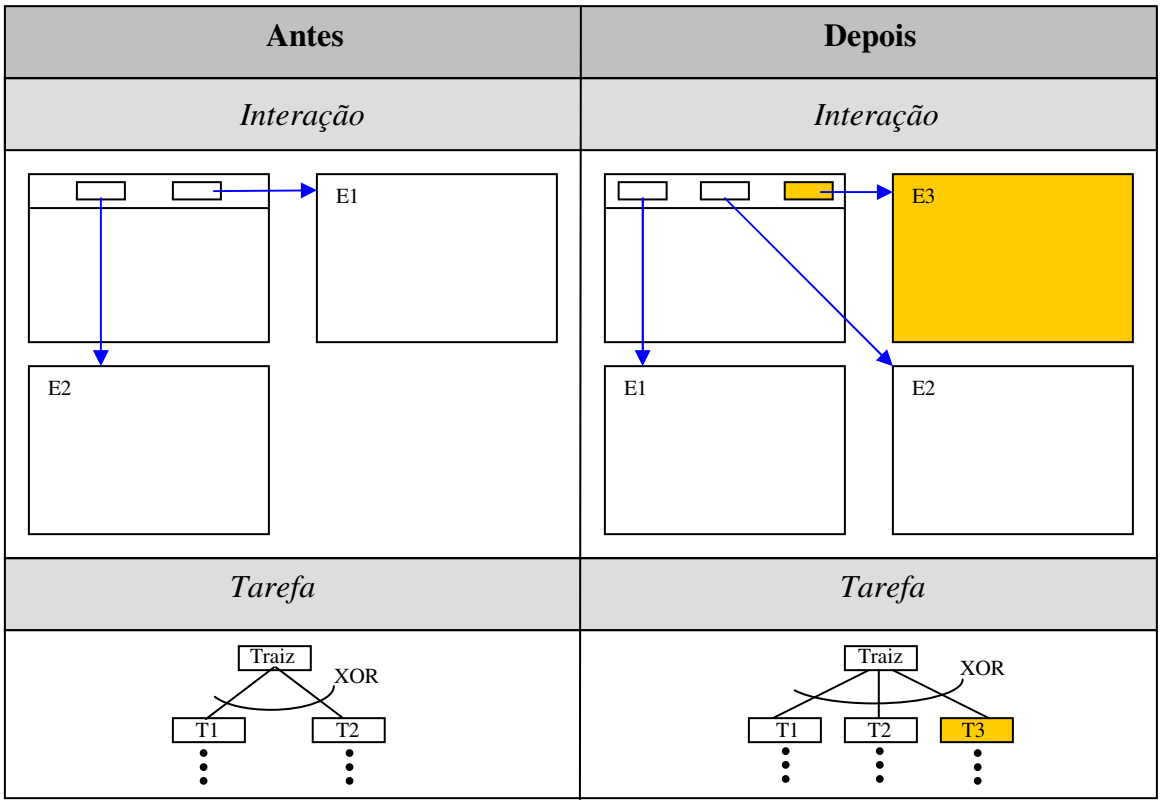
1. Introduzir o *Espaço de Interação Efilho* na árvore de Interação Abaixo do *Espaço de Direcionamento Epai*

2. Identificar o *Cenário Cpai* associado ao *Espaço de Direcionamento Epai*
3. Criar e Inserir uma *Visão Vlink* no *Epai*
4. Criar e Inserir um *Objeto de Interação OiLink* na *Vlink*
5. Criar e Inserir uma *Cena CnLink* no *Cpai*
6. Criar e Inserir uma *Tomada ToLink* na *CnLink*
7. Associar *CnLink* à *Vlink*
8. Associar *ToLink* ao *OiLink*

Criação e Mapeamento de Elementos ROTEIRO -> TAREFA

1. Identificar o *Cenário Cfilho* associado ao *Espaço de Direcionamento Efilho*
2. Introduzir o *Cenário Cfilho* na árvore de Roteiro Abaixo do *Cenário Cpai*
3. Identificar a *Tarefa Tfilha* associada ao *Cfilho*
4. Identificar a *Tarefa Tpai* associada ao *Cpai*
5. Inserir a *Tarefa Tfilha* como uma Sub-Tarefa da *Tarefa Tpai* na posição indicada
6. Modificar o atributo *Método* da *Tarefa Tpai* inserindo a nova subtarefa *Tfilha* regida pelo mesmo operador de suas irmãs (preferencialmente XOR)
7. Associar a *Cena CLink* a *Tarefa Tfilha*
8. Associar a *Tomada ToLink* a *Tarefa Tfilha*
9. Associar o *Cenário Cfilho* a *Tarefa Tfilha*

Exemplo:



1.4 - Incluir *Espaço de Direcionamento* com rearranjo (de outros *Espaços*)

Estes algoritmos têm por funcionalidade re-arranjar *Espaços* que já estejam presentes no *Modelo de Interação*. Por exemplo, se um sistema tenha funcionalidades de cadastrar produtos e usuários e os *Espaços* para cadastro de produtos e cadastro de usuários estiverem no mesmo nível, o projetista pode quer re-arranjar estes *Espaços* para que os mesmos sejam concentrados em um novo *Espaço de Direcionamento*. O re-arranjo pode ser feito a partir do *Espaço Inicial* (filho do *Espaço Inicial*) ou de um outro *Espaço de Direcionamento* (filho do *Espaço de Direcionamento*).

Assim como um Espaço de Interação, cada Espaço de Direcionamento deve ser anteriormente criado com a função **criarEspaçoDeDirecionamento.**

CriarEspaçoDeDirecionamento

Mapeamento *Elementos da INTERAÇÃO -> Elementos do ROTEIRO*

4. Criar um *Espaço de Direcionamento E*
   4.1. Criar e Inserir n *Visões V1, V2,...,Vx* no *Espaço de Direcionamento E*
   4.2. Criar e Inserir um (1) *Objeto de Interação Oix* em cada *Visão Vx* de *E*.
5. Criar um *Cenário C* e associá-lo ao *Espaço de Direcionamento E*.
   5.1. Para cada *Visão Vx* de *E* Criar e Inserir em no *Cenário C* uma *Cena Cnx*
      5.1.1. Associar cada *Visão Vx* a sua *Cena Cnx* correspondente
   5.2. Para cada *Objeto de Interação Oix* Criar e Inserir na *Cena Cnx* uma *Tomada Tox*
      5.2.1. Associar cada *Objeto de Interação Oix* a sua *Tomada Tox* correspondente

Mapeamento *Elementos do ROTEIRO -> Elementos da TAREFA*

6. Criar uma *Tarefa T* e associa-la ao *Cenário C*
   6.1. Para cada *Cena Cnx* de *C* Criar e Inserir uma *Tarefa Tfx* na *Tarefa T*
      6.1.1. Modificar o atributo *Método* de *T* para reger as suas subtarefas *Tfx* com o operador *XOR*.
      6.1.2. Associar cada *Cena Cnx* a sua *Tarefa Tfx* correspondente
      6.1.3. Associar cada *Tomada Tox* a sua *Tarefa Tfx* correspondente

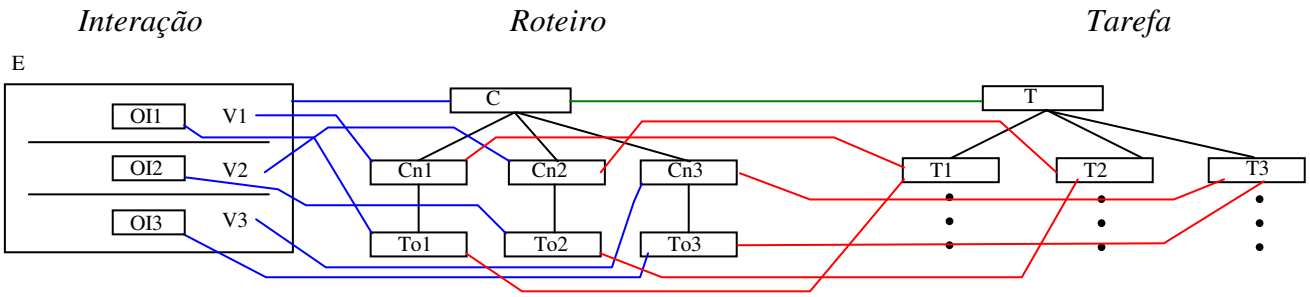


Ilustração do algoritmo de criação de Espaço de Direcionamento

1.4.1 Incluir *Espaço de Direcionamento* com rearranjo filho do *Espaço Inicial*

Descrição: Re-arranja funcionalidades (*Espaços*) a partir do *Espaço Inicial*. Retira da *Visão Funcionalidade* os *Objetos de Interação* que fazem link aos *Espaços* que estão sendo re-arranjados e insere mais um Objeto na Visão Funcionalidade para fazer link com o novo *Espaço de Direcionamento* que está sendo inserido no modelo.

Algoritmo: Inserir um *Espaço de Direcionamento* Re-arranjando Filhos do *Espaço Inicial*

1. CriarEspaçoDeDirecionamento *Efilho*

Criação e Mapeamento de Elementos INTERAÇÃO -> ROTEIRO

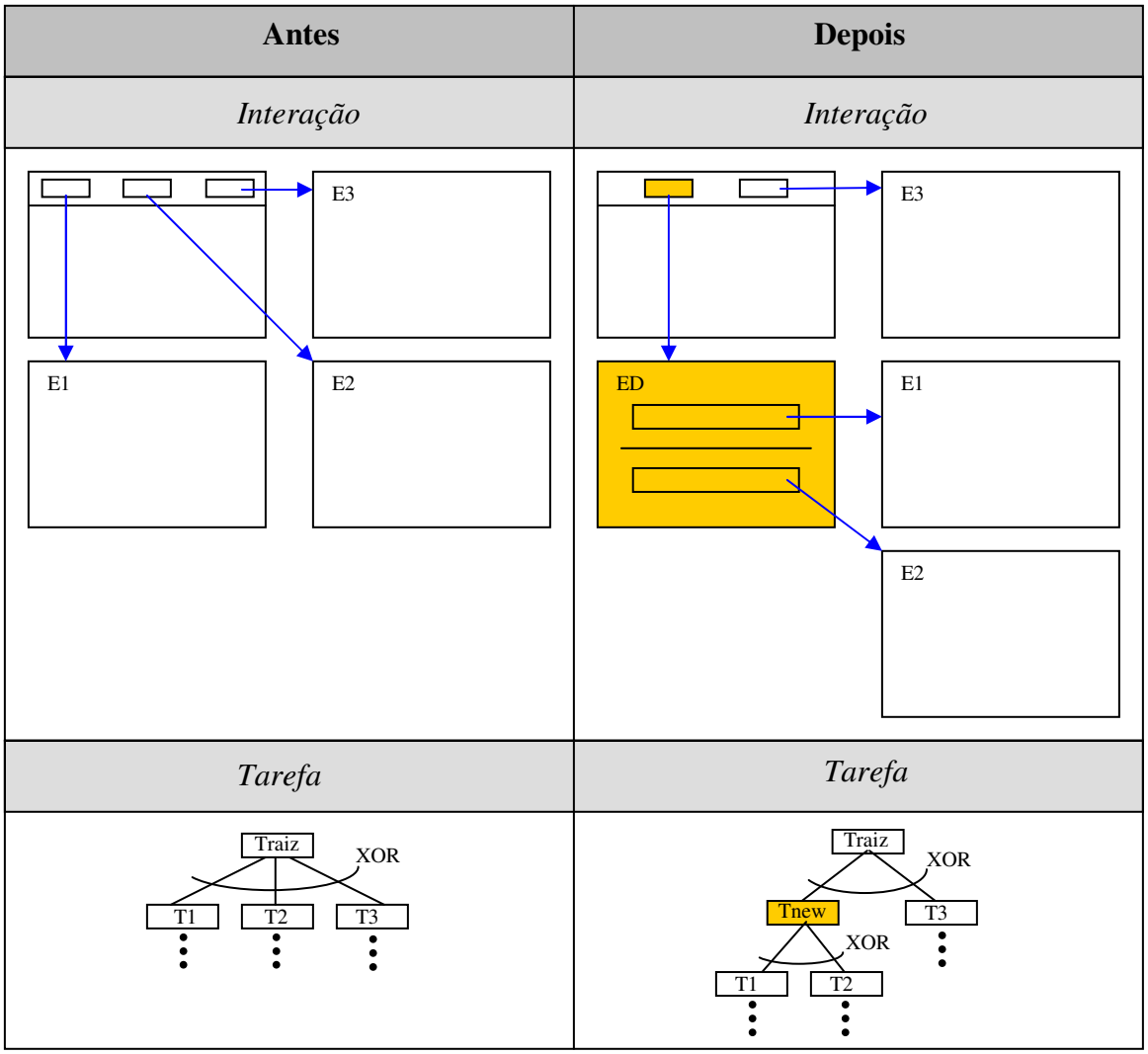
1. Identificar os *Espaços Ex* (obrigatoriamente de mesmo nível na árvore de Interação) a serem re-arranjados no *Espaço de Direcionamento Efilho*
2. Introduzir o *Espaço de Direcionamento Efilho* na árvore de Interação Abaixo do *Espaço Inicial Einicial*
3. Retirar os *Espaços Ex* da árvore de Interação e Introduzi-los novamente Abaixo do *Espaço de Direcionamento Efilho*.
4. Identificar o *Cenário Cinicial* associado ao *Einicial*
5. Identificar o *Cenário Cx* associado a cada *Espaço Ex*
6. Identificar o *Cenário Cfilho* associado ao *Efilho*
7. Introduzir o *Cenário Cfilho* na árvore de Roteiro Abaixo do *Cenário Inicial Cinicial*
8. Retirar os *Cenários Cx* da árvore de Roteiro e Introduzi-los novamente Abaixo do *Cenário Cfilho*
9. Identificar a *Visão Vfuncionalidade* do *Einicial*
10. Retirar cada *Objeto de Interação Oix* (que representada o link ao *Espaço Ex*) da *Vfuncionalidade*
11. Criar e Inserir um *Objeto de Interação OiLink* na *Vfuncionalidade*

12. Identificar a *Cena CnFuncionalidade* do *Cinicial*
13. Retirar cada *Tomada Tox* (que representada o link ao *Cenário Cx*) da *Cfuncionalidade*
14. Criar e Inserir uma *Tomada ToLink* na *Cnfuncionalidade*
15. Associar *ToLink* ao *OiLink*

Criação e Mapeamento de Elementos ROTEIRO -> TAREFA

1. Identificar a *Tarefa Tfilha* associada ao *Cenário Cfilho*
2. Identificar a *Tarefa Traiz* associada ao *Cenário Cinicial*
3. Identificar a *Tarefa Tx* associada ao *Cenário Cx*
4. Remover e Armazenar em memória as *Tarefas Tx* filhas da *Tarefa Traiz*
5. Inserir a *Tarefa Tfilha* como uma Sub-Tarefa da *Tarefa Traiz* na posição indicada
6. Modificar o atributo *Método* da *Tarefa Traiz* inserindo a nova subtarefa *Tfilha* regida pelo mesmo operador de suas irmãs (XOR)
7. Remover todas as Sub-Tarefas da *Tarefa Tfilha*
8. Inserir as Tarefas Tx (armarzenadas em memória) como Sub-tarefas da *Tarefa Tfilha*
9. Modificar o atributo *Método* da *Tarefa Traiz* inserindo as novas subtarefas *Tx* regidas pelo operador XOR
10. Associar o *Cenário Cfilho* a *Tarefa Tfilha*
11. Para cada *Tarefa Tx* Associar a *Cena CnFilhax* correspondente de *Cfilho*
12. Para cada *Tarefa Tx* Associar a Tomada *ToFilhax* correspondente de *Cnfilhax*

Exemplo:



1.4.2 Incluir *Espaço de Direcionamento* com rearranjo filho de *Espaço de Direcionamento*

Descrição: Re-arranja funcionalidades (*Espaços*) a partir de um *Espaço de Direcionamento*. Retira as *Visões e* os *Objetos de Interação* que fazem link aos *Espaços* que estão sendo re-arranjados e insere mais uma *Visão* e um *Objeto de Interação* no *Espaço*

de Direcionamento de Origem para fazer link com o novo *Espaço de Direcionamento* que está sendo inserido no modelo.

Algoritmo: Inserir um Espaço de Direcionamento Re-arranjando Filhos de outro Espaço de Direcionamento

1. CriarEspaçoDeDirecionamento *Efilho*

Criação e Mapeamento de Elementos INTERAÇÃO -> ROTEIRO

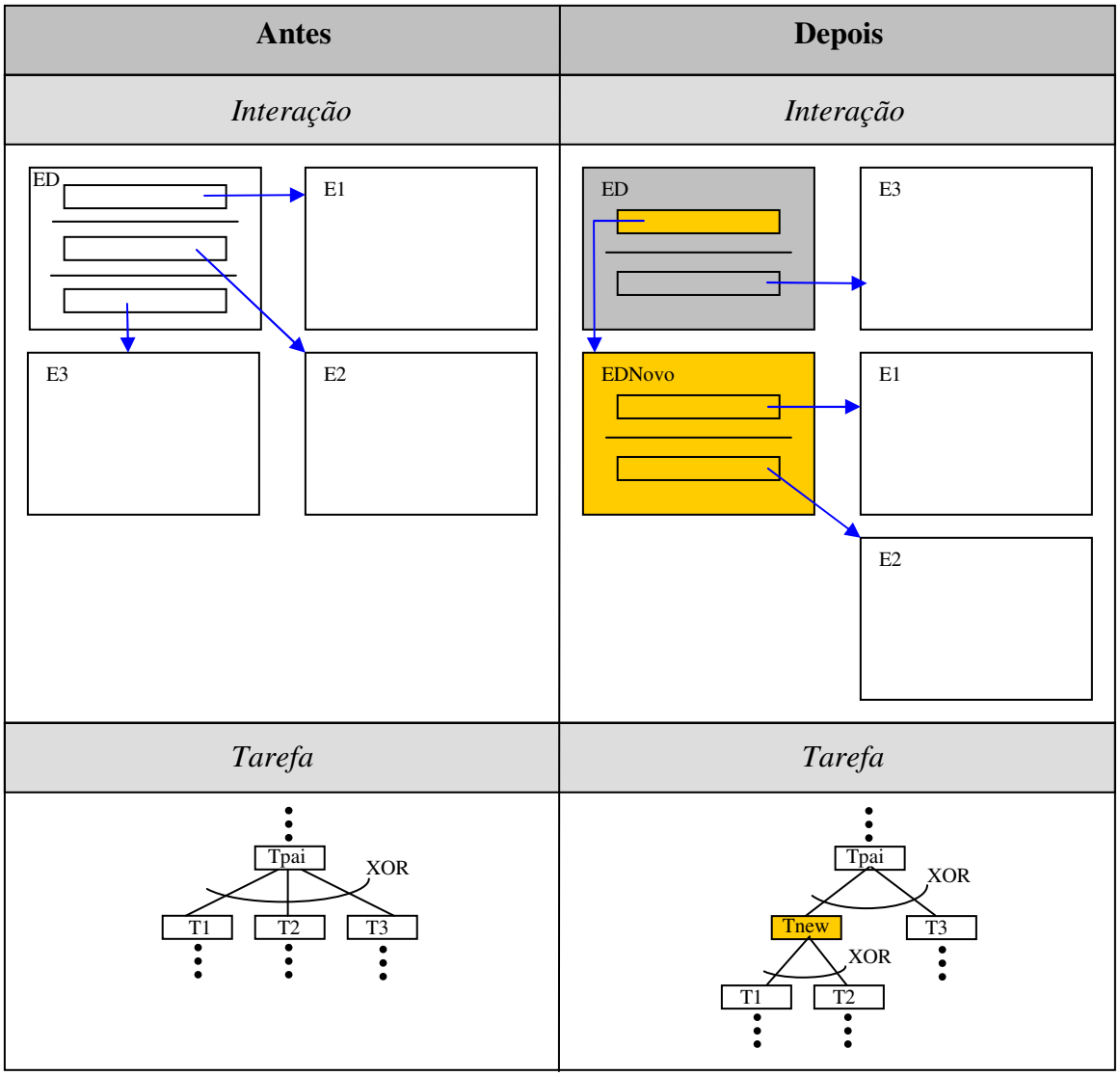
1. Identificar os *Espaços Ex* (obrigatoriamente de mesmo nível na árvore de Interação) a serem re-arranjados no *Espaço de Direcionamento Efilho*
2. Introduzir o *Espaço de Direcionamento Efilho* na árvore de Interação Abaixo do *Espaço Inicial Epai*
3. Retirar os *Espaços Ex* da árvore de Interação e Introduzi-los novamente Abaixo do *Espaço de Direcionamento Efilho*.
4. Identificar o *Cenário Cpai* associado ao *Epai*
5. Identificar o *Cenário Cx* associado a cada *Espaço Ex*
6. Identificar o *Cenário Cfilho* associado ao *Efilho*
7. Introduzir o *Cenário Cfilho* na árvore de Roteiro Abaixo do *Cenário Inicial Cpai*
8. Retirar os *Cenários Cx* da árvore de Roteiro e Introduzi-los novamente Abaixo do *Cenário Cfilho*
9. Identificar para cada *Espaço Ex* a *Visão VpaiLinkEx* do *Epai* (Esta *Visão* contém apenas um *Objeto de Interação* que representa o link ao *Espaço Ex*)
10. Retirar o *Objeto de Interação OiLinkEx* (que representada o link ao *Espaço Ex*) da *VpaiLinkEx*
11. Retirar a *Visão VpaiLinkEx* do *Espaço Epai*
12. Criar e Inserir uma Visão *VLinkEfilho* no *Espaço Epai* (esta *Visão* conterá apenas um *Objeto de Interação* que representa o link ao *Espaço de Direcionamento Efilho* que está sendo inserido)
13. Criar e Inserir um *Objeto de Interação OiLinkEfilho* na *VLinkEfilho*

14. Identificar para cada *Cenário Cx* a *Cena CnPaiLinkCx* do *Cpai* (Esta *Cena* contém apenas uma *Tomada* que representa o link ao *Cenário Cx*)
15. Retirar a *Tomada ToPaiLinkCx* (que representada o link ao *Cenário Cx*) da *CnPaiLinkx*
16. Retirar a *Cena CnPaiLinkCx* do *Cenário Cpai*
17. Criar e Inserir uma *Cena CnLinkCfilho* no *Cenário Cpai* (esta *Cena* conterá apenas uma *Tomada* que representa o link ao *Cenário Cfilho* que está sendo inserido)
18. Criar e Interir uma *Tomada ToLinkCfilho* na *Cena CnLinkCfilho*
19. Associar *ToLinkCfilho* ao *CnLinkCfilho*

Criação e Mapeamento de Elementos ROTEIRO -> TAREFA

1. Identificar a *Tarefa Tfilha* associada ao *Cenário Cfilho*
2. Identificar a *Tarefa Tpai* associada ao *Cenário Cpai*
3. Identificar as *Tarefas Tx* associada aos *Cenários Cx*
4. Remover e Armazenar em memória as *Tarefas Tx* filhas da *Tarefa Tpai*
5. Inserir a *Tarefa Tfilha* como uma Sub-Tarefa da *Tarefa Tpai* na posição indicada
6. Modificar o atributo *Método* da *Tarefa Tpai* inserindo a nova subtarefa *Tfilha* regida pelo mesmo operador de suas irmãs (XOR)
7. Remover todas as Sub-Tarefas da *Tarefa Tfilha*
8. Inserir *as Tarefas Tx* (armarzenadas em memória) como Sub-tarefas da *Tarefa Tfilha*
9. Modificar o atributo *Método* da *Tarefa Traiz* inserindo as novas subtarefas *Tx* regidas pelo operador XOR
10. Associar o *Cenário Cfilho* a *Tarefa Tfilha*
11. Para cada *Tarefa Tx* Associar a *Cena CnFilhax* correspondente de *Cfilho*
12. Para cada *Tarefa Tx* Associar a Tomada *ToFilhax* correspondente de *Cnfilhax*

Exemplo:


Antes
Depois
Interação
Interação
ED
E1
E3
E2
ED
E3
EDNovo
E1
E2
Tarefa
Tarefa
Tpai
XOR
T1
T2
T3
Tpai
XOR
Tnew
T3
XOR
T1
T2

Exclusões

Deve-se ter muito cuidado quando o projetista estiver trabalhando com as exclusões de elementos no *Modelo de Interação*. Deseja-se manter a coerência entre os modelos envolvidos em MEDITE^{+}, portanto, quando se afirma que um elemento no *Modelo de Interação* é, por exemplo, um *Espaço*, então toda a estrutura necessária para que esse elemento seja realmente um *Espaço* tem que ser mantida. É sabido que um *Espaço* contém pelo menos uma *Visão* e que esta contém pelo menos um *Objeto de Interação*. Portanto, ao se excluir elementos de um *Espaço*, há que se preservar sua estrutura mínima. Se for retirada de si essa estrutura, (fazendo, conseqüentemente, mudanças no modelo da tarefa) quando o projetista submeter o *Modelo da Tarefa,* modificado, a uma nova "roteirização", aquele elemento será re-classificado e perderá o status de *Espaço,* podendo virar qualquer outro elemento do *Modelo de Interação*. De posse dessas informações, serão definidas as *Operações de Exclusões* dos elementos do *Modelo de Interação*. Os algoritmos apresentados abaixo contêm *elementos de interação default* (vazios) apenas para a manutenção da estrutura acima relatada.

2.1 - Excluir *Objeto de interação* (se houver link, excluir também o *espaço* associado).

Descrição: Este algoritmo serve para excluir um *Objeto de Interação* de um *Espaço de Interação*. Se o *Objeto de Interação* em questão fizer link a outro *Espaço*, tal *Espaço* também deve ser excluído. Os *Objetos de Interação* de outros *Espaços* não devem ser excluídos com esse algoritmo, eles serão excluídos quando os *Espaços* a eles ligados forem excluídos.

Algoritmo: Excluir *Objeto de Interação*

1. Identificar *Espaço de Interação E (Mint.).*
2. Identificar *Visão V(Mint.).*
3. Identificar o *Objeto de Interação Oi (Mint.)* na *Visão V* do *E.I. E.*
4. SE a *Visão V* possuir três (2) ou mais *Objetos de Interação.*

4.1. ENTÃO Identificar a *Tomada To (MRo.)* associada ao *Oi (Mint.).*

4.2. Excluir *Oi(Mint.).* da *Visão V(Mint.).*

4.3. Identifique a *Ação A(MTa.)* associada à *Tomada To(MRo).*

4.4. Excluir *To (MRo.)*

4.5. Excluir a *Ação A (MTa.)* da *Tarefa P (MTa.).*

4.5.1. Modificar o atributo *Método* da *Tarefa P* excluindo *A*.

4.5.2. Extrair o diálogo *de P* através do atributo *Método* e *Ocorrência.*

5. SE a *Visão V* possuir um (1) *Objeto de Interação.* (no *Mint*. este elemento é *Visão* e *Objeto de Interação*)

5.1. ENTÃO Identificar a *Tomada To (MRo.)* associada ao *Oi (Mint.).*

5.2. Identifique a *Cena Cn (MRo.)* associado à *Visão V (Mint.).*

5.3. Identifique a *Ação A (MTa.)* associada à *Tomada To (MRo.).*

5.4. Substituir a *Visão V(Mint.) por* uma *Visão default* (vazia) *VDef (Mint.).*

5.5. Substituir *Oi (Mint.) por* um *Objeto de Interação default* (vazio) *OiDef (Mint.).*

5.6. Substituir *Cn (MRo.) por* uma *Cena default* (vazia) *CnDef (MRo.).*

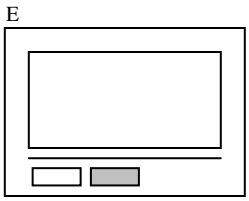
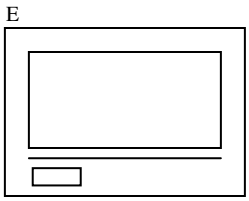
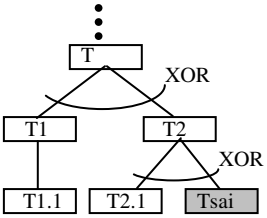
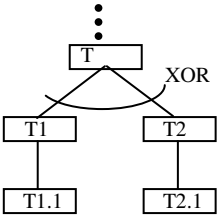
5.7. Substituir *To(MRo.) por* uma *Tomada default* (vazia) *ToDef (MRo.).*

5.8. Substituir *A (MTa.) por* uma *Ação default* (vazia) *ADef (MTa.).*

5.8.1. Modificar o atributo *Método* da *Tarefa P* substituindo *A* por *ADef*.

5.8.2. Extrair o diálogo *de P* através do atributo *Método* e *Ocorrência.*

Exemplo:

Antes	**Depois**
Interação	*Interação*
	
Tarefa	*Tarefa*
	

2.2 – Excluir Visão de Espaço de Interação

Descrição: Este algoritmo serve para excluir uma *Visão* de um *Espaço de Interação*. Se a *Visão* contiver um *Objeto de Interação* que fizer link a outro *Espaço*, tal *Espaço* também deve ser excluído.

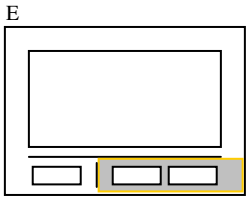
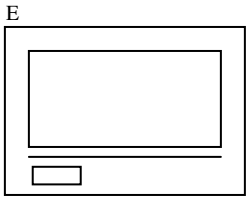
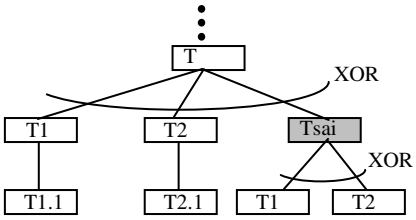
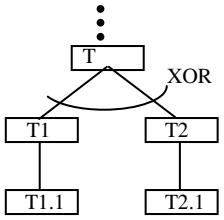
Algoritmo: Excluir Visão

1. Identificar *Espaço de Interação E (Mint.).*
2. Identificar *Visão V(Mint.)* a ser excluído de *E.*
3. SE o *Espaço de Interação E* possuir duas (2) ou mais *Visões.*
   3.1. ENTÃO Identificar a *Cena Cn (MRo.)* associada à *V (Mint.).*
   3.2. Excluir *V (Mint.).* do *Espaço de Interação E (Mint.).*
   3.3. Identifique a *Ação ou Tarefa A (MTa.)* associada à *Cena Cn (MRo).*
   3.4. Excluir *Cn (MRo.)*
   3.5. Excluir *A (MTa.)* da *Tarefa P (MTa.).*
      3.5.1. Modificar o atributo *Método* da *Tarefa P* excluindo *A*.
      3.5.2. Extrair o diálogo *de P* através do atributo *Método* e *Ocorrência.*
4. SE a *Visão V* possuir dois (2) *Objetos de Interação.*
   4.1. ENTÃO Identificar a *Cena Cn(MRo.)* associada à *V (Mint.).*
   4.2. Identifique a *Tarefa ou Ação A(MTa.)* associada à *Cena Cn(MRo).*
   4.3. Substituir *Cn (Mint.)* no *Espaço de Interação E por uma Visão default* (vazia) *VDef (Mint.).*
      4.3.1. Inserir um (1) *Objeto de interação default* (vazio) em *VDef.*
   4.4. Substituir *Cn (MRo.) por* uma *Cena default* (vazia) *CnDef (MRo.).*
      4.4.1. Inserir uma (1) *Tomada default* (vazios) em *CnDef.*
   4.5. Substituir *A (MTa.) por* uma *Tarefa default* (vazia) *TDef (MTa.).*
      4.5.1. Inserir uma *(1) Ação (MTa.) default* (vazia) em *Tdef (MTa.).*
         4.5.1.1. Modificar o atributo *Método* da *Tarefa Tdef* inserindo as *Ações Default*.

4.5.2. Modificar o atributo *Método* da *Tarefa P* substituindo *A* por *TDef*.

4.5.3. Extrair o diálogo *de P* através do atributo *Método* e *Ocorrência*

Exemplo:

Antes	**Depois**
Interação	*Interação*
	
Tarefa	*Tarefa*
	

2.3 - Excluir *Espaço de Interação*

2.3.1 - Excluir *Espaço de Interação* Filho do *Espaço Inicial*

Descrição: Este algoritmo exclui um *Espaço de Interação* diretamente do *Espaço Inicial*. Para isso, é retirado o *Objeto de Interação* da *Visão Funcionalidade* do *Espaço Inicial* que faz link ao *Espaço Interação* que está sendo excluído.

Algoritmo: Excluir um *Espaço de Interação* filho do *Espaço Inicial*

1. Identificar o *Espaço de Interação Esai* a ser Excluído do *Modelo de Interação*

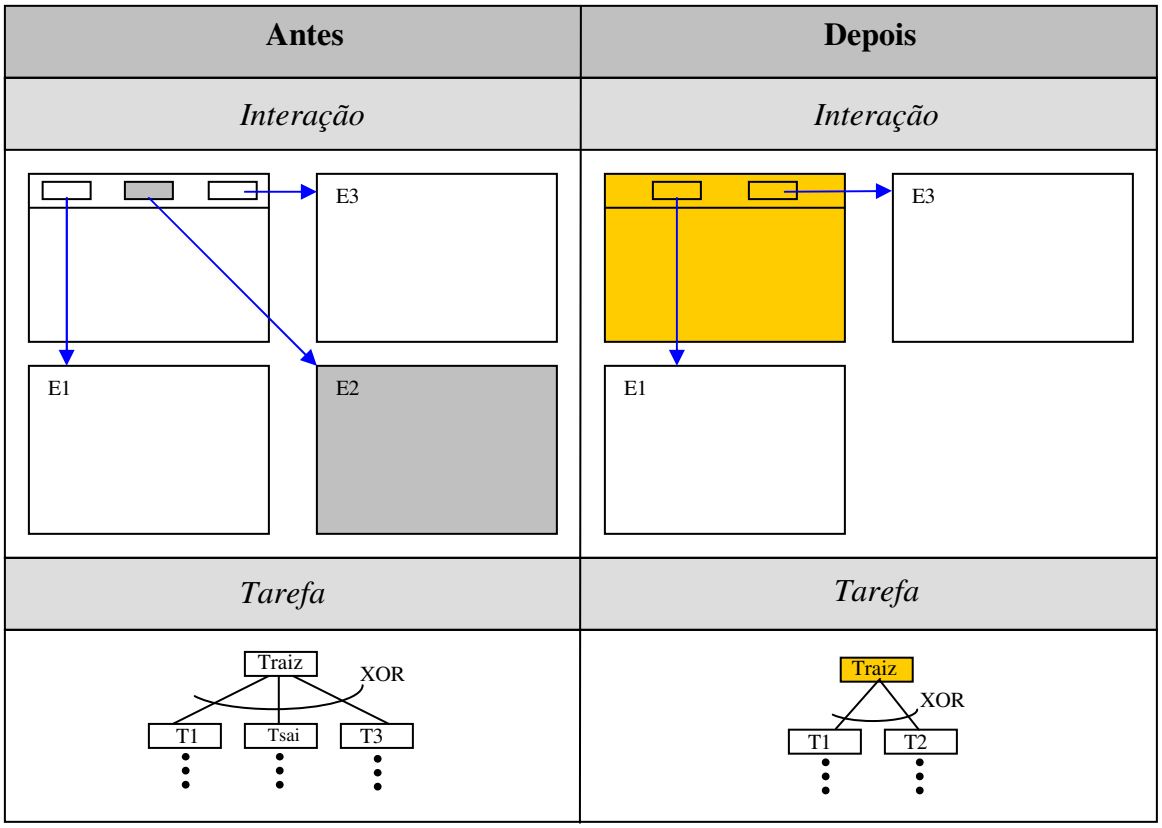
Manipulação e Mapeamento de Elementos INTERAÇÃO -> ROTEIRO

1. Identificar o *Espaço de Interação Esai* a ser retirado no *Modelo da Interação*
2. Identificar o *Espaço Inicial Einicial* que contém link ao *Esai*
3. Identificar a Visão *VFuncionalidade* do *Einicial*
4. Retirar o *Espaço de Interação Esai* da árvore de Interação
5. Identificar o *Cenário Csai* associado ao *Espaço Esai*
6. Identificar o *Cenário CInicial* associado ao *Espaço Einicial*
7. Identificar a *Cena CnFuncionalidade* associada a *Visão VFuncionalidade*
8. Retirar o *Cenário Csai* da árvore de Roteiro
9. SE a Visão VFuncionalidade possuir duas (2) ou mais Objetos de Interação ENTÃO
   - 9.1. Identificar e Retirar o *Objeto de Interação OiLinkEsai* (que representada o link ao *Espaço Esai*) da *VFuncionalidade*
   - 9.2. Identificar e Retirar a *Tomada ToLinkCsai* (que representada o link ao *Cenário Csai*) da *CnFuncionalidade*
     - 9.2.1. SENÃO
     - 9.2.2. Inserir um *Espaço EDefaultVazio* filho do *Espaço Inicial*

Manipulação e Mapeamento de Elementos ROTEIRO -> TAREFA

1. Identificar a *Tarefa Tsai* associada ao *Cenário Csai*
2. Identificar a *Tarefa TInicial* associada ao *Cenário CInicial*
3. Desassociar a *Tomada ToLinkCsai* da *Tarefa Tsai*
4. SE o *Cena CnFuncionalidade* possuir duas (2) ou mais *Tomadas* ENTÃO
5. Remover a *Tarefa Tsai* da *Tarefa TInicial*
6. Modificar o atributo *Método* da *Tarefa TInicial* retirando *Tsai*

Exemplo:



2.3.2 - Excluir *Espaço de Interação* Filho de *Espaço de Direcionamento*

Descrição: Este algoritmo exclui um *Espaço de Interação* diretamente de um *Espaço de Direcionamento*. Para isso, a *Visão* e o *Objeto de Interação* que faz link ao *Espaço Interação* que está sendo excluído são excluídos do *Espaço de Direcionamento.*

Algoritmo: Excluir um *Espaço de Interação* filho de um *Espaço de Direcionamento*

1. Identificar o *Espaço de Interação Esai* a ser Excluído do *Modelo de Interação*

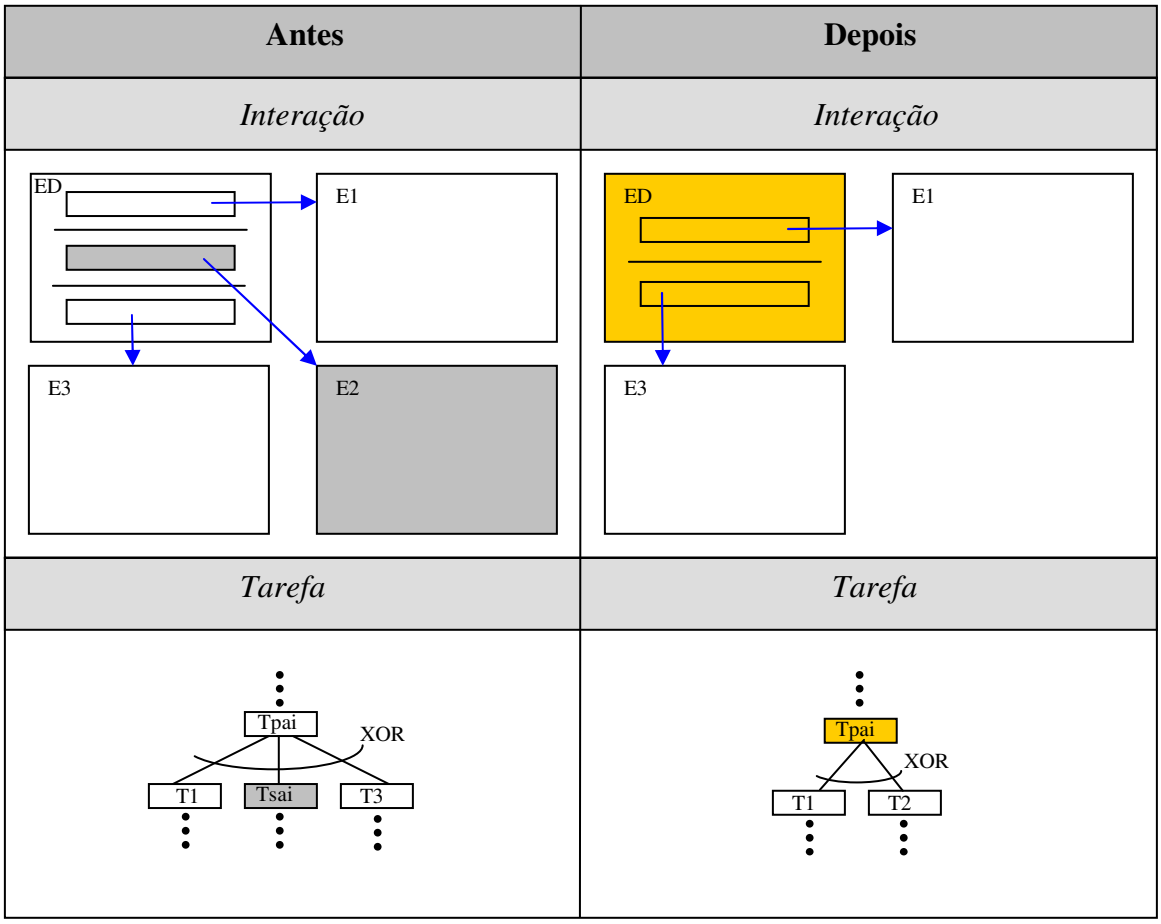
Manipulação e Mapeamento de Elementos INTERAÇÃO -> ROTEIRO

1. Identificar os *Espaço de Interação Esai* a ser retirado no *Modelo da Interação*
2. Identificar o *Espaço de Direcionamento Epai* que contém link ao *Esai*
3. Retirar o *Espaço de Interação Esai* da árvore de Interação
4. Identificar o *Cenário Csai* associado ao *Espaço Esai*
5. Identificar o *Cenário Cpai* associado ao *Espaço Epai*
6. Retirar o *Cenário Csai* da árvore de Roteiro
7. SE o Espaço *Epai* possuir duas (2) ou mais *Visões* ENTÃO
   - 7.1. Identificar e Retirar a *Visão VLinkEsai* do *Epai*
   - 7.2. Retirar o *Objeto de Interação OiLinkEsai* (que representada o link ao *Espaço Esai*) da *VlinkEsai*
   - 7.3. Identificar e Retirar a *Cena CnLinkCsai* do *Cdestino*
   - 7.4. Retirar a *Tomada ToLinkCsai* (que representada o link ao *Cenário Csai*) da *CnLinkCsai*
     - 7.4.1. SENÃO Substituir *VlinkEsai* por uma *Visão VdefaultVazia*
     - 7.4.2. Substituir *OiLinkEsai* por um Objeto de Interação *OiDefaultVazio*
     - 7.4.3. Substituir *CnLinkCsai* por uma *Cena CndefaultVazia*
     - 7.4.4. Substituir *ToLinkCsai* por um Objeto de Interação *ToDefaultVazia*

Manipulação e Mapeamento de Elementos ROTEIRO -> TAREFA

1. Identificar a *Tarefa Tsai* associada ao *Cenário Csai*
2. Identificar a *Tarefa Tpai* associada ao *Cenário Cpai*
3. Desassociar a *Cena CnLinkEsai* da *Tarefa Tsai*
4. Desassociar a *Tomada ToLink* da *Tarefa Tsai*
5. SE o Cenário *Cpai* possuir duas (2) ou mais *Cenas* ENTÃO
   5.1. Remover a *Tarefa Tsai* da *Tarefa Tpai*
   5.2. Modificar o atributo *Método* da *Tarefa Tpai* retirando *Tsai*
      5.2.1. SENÃO Inserir uma *Tarefa TdefaultVazia* como filha de *Tpai*
      5.2.2. Associar a *Tarefa TdefaultVazia* à *Cena CndefaultVazia*
      5.2.3. Inserir uma *Ação AdefaultVazia* como filha da *Tarefa TdefaultVazia*
      5.2.4. Associar *ToDefaultVazia* à *Ação AdefaultVazia*

Exemplo:



2.3.3 - Excluir *Espaço de Interação* Filho de *Espaço de Interação*

Descrição: Este algoritmo exclui um *Espaço de Interação* diretamente de outro *Espaço de Interação*. Para isso, a *Visão* e o *Objeto de Interação* do *Espaço de Interação de Origem* que faz link ao *Espaço de Interação* que está sendo excluído também são excluídos.

Algoritmo: Excluir um *Espaço de Interação* filho de um *Espaço de Interação*

1. Identificar o *Espaço de Interação Esai* a ser Excluído do *Modelo de Interação*

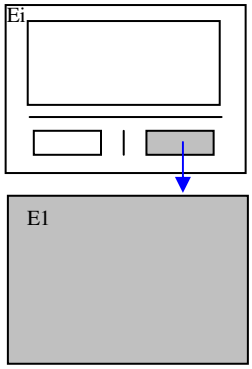
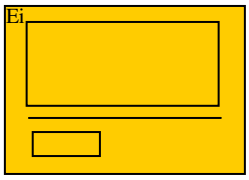
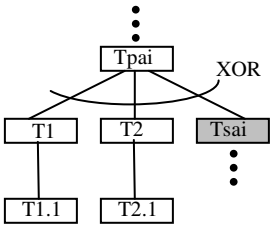
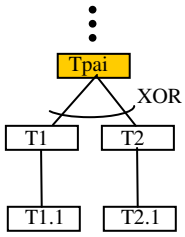
Manipulação e Mapeamento de Elementos INTERAÇÃO -> ROTEIRO

1. Identificar os *Espaço de Interação Esai* a ser retirado no *Modelo da Interação*
2. Identificar o *Espaço de Direcionamento Epai* que contém link ao *Esai*
3. Retirar o *Espaço de Interação Esai* da árvore de Interação
4. Identificar o *Cenário Csai* associado ao *Espaço Esai*
5. Identificar o *Cenário Cpai* associado ao *Espaço Epai*
6. Retirar o *Cenário Csai* da árvore de Roteiro
7. SE o Espaço *Epai* possuir duas (2) ou mais *Visões* ENTÃO
   - 7.1. Identificar e Retirar a *Visão VLinkEsai* do *Epai*
   - 7.2. Retirar o *Objeto de Interação OiLinkEsai* (que representada o link ao *Espaço Esai*) da *VlinkEsai*
   - 7.3. Identificar e Retirar a *Cena CnLinkCsai* do *Cdestino*
   - 7.4. Retirar a *Tomada ToLinkCsai* (que representada o link ao *Cenário Csai*) da *CnLinkCsai*
     - 7.4.1. SENÃO Substituir *VlinkEsai* por uma *Visão VdefaultVazia*
     - 7.4.2. Substituir *OiLinkEsai* por um Objeto de Interação *OiDefaultVazio*
     - 7.4.3. Substituir *CnLinkCsai* por uma *Cena CndefaultVazia*
     - 7.4.4. Substituir *ToLinkCsai* por um Objeto de Interação *ToDefaultVazia*

Manipulação e Mapeamento de Elementos ROTEIRO -> TAREFA

1. Identificar a *Tarefa Tsai* associada ao *Cenário Csai*
2. Identificar a *Tarefa Tpai* associada ao *Cenário Cpai*
3. Desassociar a *Cena CnLinkEsai* da *Tarefa Tsai*
4. Desassociar a *Tomada ToLink* da *Tarefa Tsai*
5. SE o Cenário *Cpai* possuir duas (2) ou mais *Cenas* ENTÃO
   5.1. Remover a *Tarefa Tsai* da *Tarefa Tpai*
   5.2. Modificar o atributo *Método* da *Tarefa Tpai* retirando *Tsai*
      5.2.1. SENÃO Inserir uma *Tarefa TdefaultVazia* como filha de *Tpai*
      5.2.2. Associar a *Tarefa TdefaultVazia* à *Cena CndefaultVazia*
      5.2.3. Inserir uma *Ação AdefaultVazia* como filha da *Tarefa TdefaultVazia*
      5.2.4. Associar *ToDefaultVazia* à *Ação AdefaultVazia*

Exemplo:

Antes	**Depois**
Interação	*Interação*
	
Tarefa	*Tarefa*
	

2.4 - Excluir *Espaço de Direcionamento* (sem excluir seus filhos)

2.4.1 - Excluir *Espaço de Direcionamento* Re-arranjando filhos no Espaço Inicial

Descrição: Exclui um *Espaço de Direcionamento* re-arranjando seus filhos no *Espaço Inicial*. Esta operação é o inverso da inclusão de *Espaço de Direcionamento* re-arranjando seus filhos no *Espaço Inicial.*

Algoritmo: Excluir um *Espaço de Direcionamento* Re-arranjando Filhos no *Espaço Inicial*

1. Identificar o *Espaço de Direcionamento Esai* a ser Excluído do *Modelo de Interação*

Exclusão e Mapeamento de Elementos INTERAÇÃO -> ROTEIRO

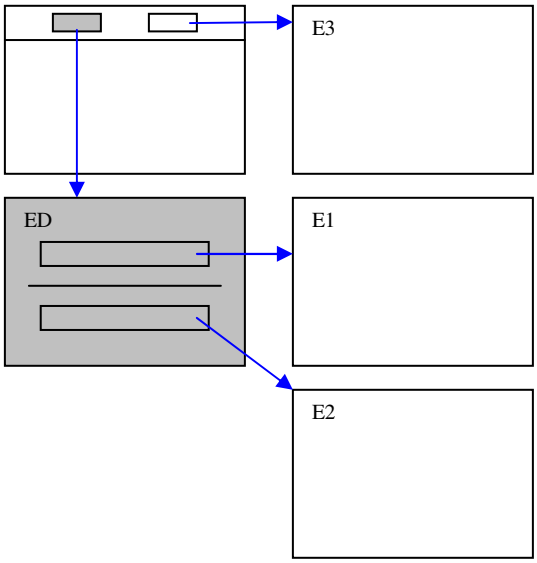
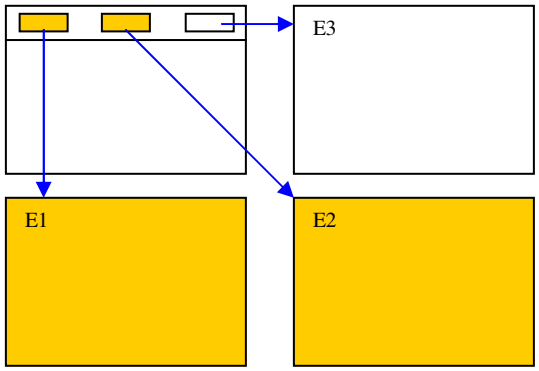
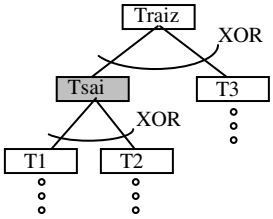
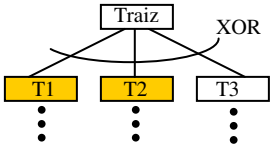
1. Identificar os *Espaços Ex* (obrigatoriamente de mesmo nível na árvore de Interação) a serem re-arranjados no *Espaço Inicial Einicial*
2. Retirar os *Espaços Ex* da árvore de Interação e Introduzi-los novamente Abaixo do *Espaço de Inicial Einicial*.
3. Retirar o *Espaço de Direcionamento Esai* da árvore de Interação
4. Identificar o *Cenário Cinicial* associado ao *Einicial*
5. Identificar o *Cenário Cx* associado a cada *Espaço Ex*
6. Identificar o *Cenário Csai* associado ao *Esai*
7. Retirar os *Cenários Cx* da árvore de Roteiro e Introduzi-los novamente Abaixo do *Cenário Cinicial*
8. Retirar o *Cenário Csai* da árvore de Roteiro
9. Identificar a *Visão Vfuncionalidade* do *Einicial*
10. Retirar o *Objeto de Interação OiLinkSai* (que representada o link ao *Espaço Esai*) da *Vfuncionalidade*
11. Criar e Inserir para cada *Espaço Ex* um *Objeto de Interação OiLinkx* na *Vfuncionalidade*
12. Identificar a *Cena CnFuncionalidade* do *Cinicial*

13. Retirar a *Tomada Tox* (que representada o link ao *Cenário Csai*) da *Cnfuncionalidade*
14. Criar e Inserir para cada *Cenário Cx* uma *Tomada ToLinkx* na *Cnfuncionalidade*
15. Associar *ToLinkx* ao *OiLinkx Respectivo*

Exclusão e Mapeamento de Elementos ROTEIRO -> TAREFA

1. Identificar a *Tarefa Tsai* associada ao *Cenário Csai*
2. Identificar a *Tarefa Traiz* associada ao *Cenário Cinicial*
3. Identificar a *Tarefa Tx* associada ao *Cenário Cx*
4. Remover e Armazenar em memória as *Tarefas Tx* filhas da *Tarefa Tsai*
5. Remover a *Tarefa Tsai* como uma Sub-Tarefa da *Tarefa Traiz*
6. Inserir as *Tarefas Tx* (armarzenadas em memória) como Sub-tarefas da *Tarefa Traiz*
7. Modificar o atributo *Método* da *Tarefa Traiz* inserindo as novas subtarefas *Tx* regidas pelo operador XOR
8. Para cada *Tarefa Tx* Desassociar a *Cena CnSaix* correspondente de *Csai*
9. Para cada *Tarefa Tx* Desassociar a Tomada *ToSaix* correspondente de *CnSaix*

Exemplo:

Antes	Depois
Interação	*Interação*
	
Tarefa	*Tarefa*
	

2.4.2 - Excluir *Espaço de Direcionamento* Re-arranjando filhos em outro Espaço de Direcionamento

Descrição: Exclui um *Espaço de Direcionamento* re-arranjando seus filhos em outro *Espaço de Direcionamento*. Esta operação é o inverso da inclusão de *Espaço de Direcionamento* re-arranjando seus filhos em outro *Espaço de Direcionamento.*

Algoritmo: Excluir um *Espaço de Direcionamento* Re-arranjando Filhos em outro *Espaço de Direcionamento*

1. Identificar o *Espaço de Direcionamento Esai* a ser Excluído do *Modelo de Interação*

Exclusão e Mapeamento de Elementos INTERAÇÃO -> ROTEIRO

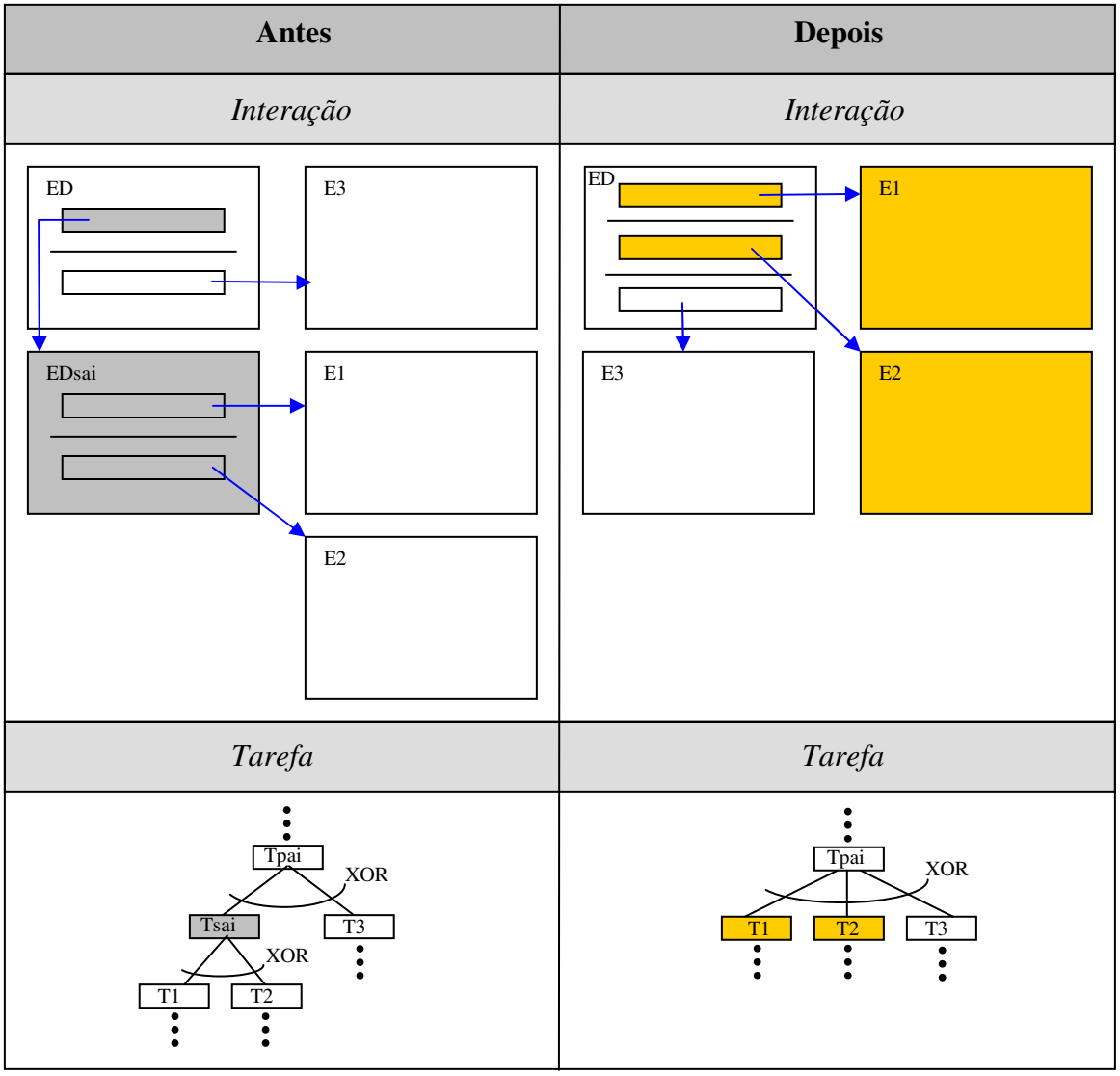
1. Identificar os *Espaços Ex* (obrigatoriamente de mesmo nível na árvore de Interação) a serem re-arranjados no *Espaço de Direcionamento Edestino*
2. Retirar os *Espaços Ex* da árvore de Interação e Introduzi-los novamente Abaixo do *Espaço de Direcionamento Edestino*.
3. Retirar o *Espaço de Direcionamento Esai* da árvore de Interação
4. Identificar o *Cenário Cdestino* associado ao *Edestino*
5. Identificar o *Cenário Cx* associado a cada *Espaço Ex*
6. Identificar o *Cenário Csai* associado ao *Esai*
7. Retirar os *Cenários Cx* da árvore de Roteiro e Introduzi-los novamente Abaixo do *Cenário Cdestino*
8. Retirar o *Cenário Csai* da árvore de Roteiro
9. Identificar e Retirar a *Visão VLinkEsai* do *Edestino*
10. Retirar o *Objeto de Interação OiLinkEsai* (que representada o link ao *Espaço Esai*) da *VLinkEsai*
11. Criar e Inserir para cada *Espaço Ex* uma *Visão VlinkEx* no *Edestino*
    11.1. Criar e Inserir em *VLinkEx* um *Objeto de Interação OiLinkEx*
12. Identificar e Retirar a *Cena CnLinkCsai* do *Cdestino*

13. Retirar a *Tomada ToLinkCsai* (que representada o link ao *Cenário Csai*) da *CnLinkCsai*
14. Criar e Inserir para cada *Cenário Cx* uma *Cena CnLinkCx no Cenário Cdestino*
    14.1. Criar e Inserir em *CnLinkCx uma Tomada ToLinkCx*
15. Associar *CnLinkCx* a *VlinkEx* Respectio
16. Associar *ToLinkCx* ao *OiLinkEx* Respectivo

Exclusão e Mapeamento de Elementos ROTEIRO -> TAREFA

1. Identificar a *Tarefa Tsai* associada ao *Cenário Csai*
2. Identificar a *Tarefa Tdestino* associada ao *Cenário Cdestino*
3. Identificar a *Tarefa Tx* associada ao *Cenário Cx*
4. Remover e Armazenar em memória as *Tarefas Tx* filhas da *Tarefa Tsai*
5. Remover a *Tarefa Tsai* como uma Sub-Tarefa da *Tarefa Tdestino*
6. Inserir as *Tarefas Tx* (armarzenadas em memória) como Sub-tarefas da *Tarefa Tdestino*
7. Modificar o atributo *Método* da *Tarefa Tdestino* inserindo as novas subtarefas *Tx* regidas pelo operador XOR
8. Para cada *Tarefa Tx* Desassociar a *Cena CnSaix* correspondente de *Csai*
9. Para cada *Tarefa Tx* Desassociar a Tomada *ToSaix* correspondente de *CnSaix*
10. Para cada *Tarefa Tx* Associar a *Cena CnLinkEx* correspondente de *Cdestino*
11. Para cada *Tarefa Tx* Associar a *Tomada ToLinkEx* correspondente de *Cndestinox*

Exemplo:


Antes
Depois
Interação
Interação
ED
E3
EDsai
E1
E2
ED
E1
E3
E2
Tarefa
Tarefa
Tpai
XOR
Tsai
T3
XOR
T1
T2
Tpai
XOR
T1
T2
T3