UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

SOFIA LARISSA DA COSTA

Uma Abordagem Baseada em Modelos para Construção Automática de Interfaces de Usuário para Sistemas de Informação

Goiânia
2011

SOFIA LARISSA DA COSTA

Uma Abordagem Baseada em Modelos para Construção Automática de Interfaces de Usuário para Sistemas de Informação

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Computação.

Área de concentração: Sistemas de Informação.

Orientador: Prof. Juliano Lopes de Oliveira

Goiânia
2011

Sofia Larissa da Costa

Graduou-se em Ciência da Computação na UFT - Fundação Universidade Federal do Tocantins. Durante sua graduação, foi Desenvolvedora *Web* da Tecnoplace Gestão e Tecnologia. Durante o mestrado na UFG, foi bolsista da CAPES e desenvolveu pesquisa no projeto Estratégias e Métodos para Melhorias de Software.

À Deus, sem a qual não posso viver.
Aos meus pais, pelo incentivo e apoio incondicional.
Ao Glesio, pelo amor e compreensão.

Agradecimentos

A Deus, por ter me dado a oportunidade de concluir esta etapa. Por ter me iluminado e concedido sabedoria. Sem Ele nada posso.

Aos meus pais, Abner e Jeane, pelo amor e apoio em todos os momentos. Pelas conversas ao fim do dia, pela preocupação e pelos momentos de distração. Na distância em que estivemos nestes dois anos pude perceber o quanto são valiosos em minha vida, obrigada por tudo!

A minha irmã Deyse, que mesmo distante se fez presente, pela atenção e carinho em todos os momentos.

Ao Glesio, pelo amor, carinho, atenção, apoio e compreensão durante estes dois anos. Agradeço pelo incentivo! Por sua família que me acolheu em muitos momentos.

Às minhas tias Elenice e Noemi, por terem me acolhido nestes dois anos, obrigada pela amizade, paciência e atenção.

Ao meu orientador, professor Juliano, pela paciência e orientação, por ter transmitido conhecimentos que serão importantes durante minha vida profissional, e por ser um exemplo para cada um de seus alunos na formação de novos conhecimentos.

Aos meus colegas do grupo de pesquisa do Mestrado, Valdemar e Luiz, que auxiliaram nos momentos de estudo e pela amizade. Aos colegas de mestrado, Fabiana, Patrícia, Marcelo, Adriana, Elisabete e Daniel por terem caminhado junto comigo nesta jornada, pelos conhecimentos partilhados e pelos momentos de distração.

Aos professores do INF, pelos ensinamentos e auxílio. Aos funcionários do mestrado do INF, pelos trabalhos realizados com presteza.

À CAPES, pelo apoio financeiro.

"Porque o SENHOR dá a sabedoria; da sua boca é que vem o conhecimento e o entendimento."

Provérbios 2:6 ,
Bíblia Sagrada, Edição Almeida Corrigida e Revisada.

Resumo

Da Costa, Sofia Larissa. **Uma Abordagem Baseada em Modelos para Construção Automática de Interfaces de Usuário para Sistemas de Informação** . Goiânia, 2011. 114p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

A construção de interfaces de usuário para Sistemas de Informação (SI) envolve modelagem e codificação de aspectos de aparência (apresentação) e comportamento (forma de interação). Este trabalho propõe uma abordagem baseada em modelos para construção dessas interfaces com o apoio de ferramentas de transformação automática de modelos e de geração de código de interface. A abordagem utiliza o conceito de Estereótipo de Interface, introduzido neste trabalho, que identifica, em alto nível de abstração, características de aparência e comportamento de interfaces, independentemente da aplicação do Sistema de Informação subjacente. Uma taxonomia para elementos de interface é proposta como base para a definição de estereótipos, juntamente com um mecanismo para especificação do comportamento da interface, que permite expressar ações e restrições sobre estereótipos de interface de maneira precisa, objetiva e independente da plataforma de implementação da interface. Também é proposta uma arquitetura para um componente de software que gerencia a construção de interfaces baseada em modelos. A arquitetura define como este componente pode ser integrado ao processo de desenvolvimento de SI. A abordagem para construção baseada em modelos proposta neste trabalho traz benefícios em termos de esforço e custo de construção facilitando a manutenção e a evolução de interfaces de usuário em SI. Além disso, o uso de estereótipos promove a consistência e a padronização, tanto da apresentação quanto do comportamento das interfaces, melhorando a usabilidade de SI.

Palavras–chave

Interfaces de Usuário para Sistemas de Informação; Modelagem de Interface de Usuário; Estereótipo de Interface; Construção Automática de Interfaces de Usuário baseada em Modelos.

Abstract

Da Costa, Sofia Larissa. **A Model-Based Approach to User Interfaces Automatic Building for Information Systems**. Goiânia, 2011. 114p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

Building user interfaces for Information Systems (IS) involves modeling and coding appearance (presentation) and behavioral (interaction) aspects. This work presents a model-based approach to building these interfaces using tools for automatic transformation of models and for interface code generation. The proposed approach applies the concept of Interface Stereotype, introduced in this work, which identifies, in a high level of abstraction, features of user interface (UI) appearance and behavior, independently of the underlying IS application. A taxonomy of interface elements is proposed as the basis for stereotype definition, along with a interface behavior specification mechanism, which allows expressing actions and restrictions on the stereotypes by precise, objective and independently from the interface implementation platform. It is also proposed a architecture for a software component which manages model-based user interfaces building. The architecture defines how this component can be integrated in IS development process. The approach for model-based user interface development proposed in this work brings benefits in effort and cost construction terms, facilitating the maintenance and the evolution of user interface of IS. Futhermore, the use of stereotypes promotes consistency and standardization of both presentation and behavior of interfaces, improving usability of IS.

Keywords

User interface of Information Systems; User interface modeling; Interface Stereotype; Model-based User Interface automatic building.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Interfaces de Usuário (IU) são componentes que permitem a interação entre pessoas e um sistema computacional [39]. Apesar de significativos avanços obtidos pela comunidade de pesquisa em IU, construir interfaces de alta qualidade é um desafio ainda presente [30].

Abordagens automatizadas para construção de IU podem ajudar a enfrentar esse desafio, trazendo benefícios não apenas em termos de produtividade, mas também em relação à qualidade das IU geradas, além do maior controle para sua evolução.

Este trabalho propõe uma abordagem baseada em modelos para automatizar a construção de IU para Sistemas de Informação (SI), isto é, sistemas que possuem coleções de dados e de processos que coletam, armazenam, transformam e distribuem informações por meio de uma interface gráfica [63].

1.1 Dificuldades para Geração Automática de IU

A geração de IU para aplicações interativas de SI engloba a modelagem de tarefas e informações que o usuário necessita manipular, dos relacionamentos entre tarefas e informações, e da forma utilizada pelo usuário para realizar cada tarefa. A complexidade dessa modelagem, e da implementação dos modelos resultantes, torna o processo de construção de IU para SI difícil, caro e propenso a erros. De fato, questões relativas a IU podem consumir mais de 50% do tempo de desenvolvimento de SI [30].

Há duas abordagens principais para desenvolvimento de IU [18]: o mapeamento manual das necessidades do usuário para linguagens computacionais; e a construção automatizada a partir de modelos. O foco deste trabalho está nesta última abordagem.

O desenvolvimento de IU baseado em modelos [66, 9], é definido como o processo de criar e refinar modelos visando a geração de IU. O uso de modelos aumenta o nível de abstração, ajudando o usuário a entender com mais clareza como os requisitos são transformados em interfaces, e também facilita a reutilização de conceitos de interfaces em outros projetos [1, 37]. Esta abordagem tem sido foco de muitas pesquisas que resultaram no desenvolvimento de diversas ferramentas [58, 66].

Esforços recentes para construção de IU baseada em modelos têm focado na modelagem em alto nível de abstração e na geração automática do código, de forma a modelar IU sem a dependência de um especialista em usabilidade e a obter maior eficiência que a programação manual de IU [22].

Apesar dos avanços obtidos, ainda não é possível criar automaticamente soluções completas para IU [72, 76, 38, 45]. Por exemplo, um dos princípios de projeto de IU é a separação entre o código de interface e a lógica da aplicação [44], mas este princípio não tem sido considerado pelas abordagens de geração de IU. Em muitas abordagens, o design de IU ainda é um subproduto acidental ou oportunístico da construção de software.

Mesmo com a existência de abordagens automatizadas, grande parte das IU ainda é desenvolvida com construção manual de código e utilização de modelos de baixo nível por desenvolvedores que não conhecem o domínio da aplicação e as necessidades dos usuários [4].

A qualidade das IU geradas ainda é inferior à das manufaturadas, e a necessidade de personalização das IU geradas contribui para dificultar este problema. Outros desafios na construção de IU são a modelagem de um vasto conjunto de eventos relacionados ao comportamento de IU; a falta de apoio à decomposição dos elementos da interface, a fim de serem reutilizados; e a dificuldade de integração de IU geradas com a lógica de negócio de SI subjacentes.

Nesse sentido, buscou-se compreender os desafios existentes no desenvolvimento de IU baseado em modelos, de forma a compreender quais problemas devem ser atacados na evolução das atuais abordagens.

A Tabela 1.1 sumariza os desafios identificados nesta seção. Diante da diversidade de trabalhos com abordagens para o desenvolvimento de IU baseado em modelos, é um desafio identificar qual deles possui as propriedades necessárias para construir IU para SI. Antes disso, torna-se indispensável extrair um conjunto de características que descrevam aspectos de apresentação e comportamento de IU bem como sobrelevem os desafios mencionados.

A Engenharia de IU requer uma linguagem estruturada para a descrição de IU (desafio 1 da Tabela 1.1). Isto significa que os modelos devem ser descritos em uma linguagem não-ambígua e visual de forma que seja possível compreender quais requisitos devem ser considerados na descrição de IU. Isto também torna possível o compartilhamento das informações dos modelos entre diferentes ferramentas [72].

Ferramentas para o desenvolvimento de IU baseado em modelos tentam produzir automaticamente uma interface de usuário concreta a partir de um modelo abstrato. Em geral, elas atingem esse propósito com um sucesso limitado. A razão para esta limitação tem sido identificada como o *problema do mapeamento* [56] (desafio 2), que consiste na dificuldade de criar uma ligação entre elementos abstratos e elementos concretos dentro

Tabela 1.1: *Desafios para Construção de IU Baseada em Modelos*

Número do Desafio	**Desafios**
1	Linguagem Estruturada para descrição de IU
2	Problema do Mapeamento
3	Processo de Embelezamento
4	Especificação do Leiaute
5	Informações dos conceitos do domínio para a construção de IU
6	Apoio à decomposição de IU
7	Separação de preocupações
8	Correlação dos modelos envolvidos
9	Integração de IU com os demais aspectos de SI
10	Integração de padrões de IHC e desenvolvimento de IU baseada em modelos

de um modelo de interfaces. Esse problema resulta da imprecisa compreensão de quais características e atributos dos elementos abstratos governam a criação dessas ligações [33].

As transformações de um modelo abstrato para as IU propriamente ditas têm um sucesso limitado, devido à dificuldade do mapeamento entre os diferentes níveis de abstração, causada pela falta de compreensão sobre as ligações entre as características dos modelos-origem e dos modelos-destino [56].

Os modelos de apresentação, em geral, apenas esboçam o leiaute de IU, e não fornecem recursos suficientes para refinar a interface. Esta preocupação emerge quando surge a necessidade de lidar com os gostos e preferências dos usuários, ou para seguir alguma orientação de estilo, que são preocupações do *processo de embelezamento* [38, 72, 52] (desafio 3).

Outra dificuldade para a construção de IU baseada em modelos consiste na especificação de leiaute (desafio 4), que é considerado um problema complexo [76]. Outras informações, como objetos de dados, seus tipos, domínio dos valores, relacionamentos, restrições operacionais, entre outros fatores, têm alguma contribuição para a constituição de IU. Mas estas informações ainda não tem sido completamente enfatizadas e discutidas antes da modelagem de IU [76, 72] (desafio 5).

Apoiar a decomposição de IU, disponibilizando os elementos de IU existentes para outros projetos é outro desafio na construção de IU (desafio 6). Com a superação deste desafio serão alcançados benefícios em termos de reusabilidade e produtividade no desenvolvimento de IU. As atuais técnicas de modelagem de IU carecem de um mecanismo intuitivo para descrever diretamente a composição e apresentação da interface com os conceitos abstratos relacionados [72].

O princípio de separação de preocupações para a modelagem de um sistema

(desafio 7) representa um desafio para a abordagem baseada em modelos, uma vez que a maioria dos conceitos de IHC não pode ser facilmente descrito independentemente de outros conceitos do domínio do sistema e, geralmente, cada aspecto do sistema tem um impacto sobre os usuários. Assim, os modelos devem englobar completa e separadamente a estrutura e o comportamento da interface, sem considerar aspectos pertinentes ao domínio da aplicação [27, 72].

A própria complexidade dos modelos tem afetado a usabilidade das abordagens existentes, causando dificuldade para identificar o subconjunto de modelos que se aplicam em cada caso (desafio 8). Ainda mais, quanto mais modelos forem aplicados, maior será a correlação entre os modelos, enquanto é mantida sua separação. Isto pode resultar na redução da atratividade e viabilidade da metodologia por completo, já que a aplicação de um certo subconjunto de modelos não garante que todos os detalhes necessários ao funcionamento do Sistema de Informação serão gerados [72].

Mas, ao mesmo tempo que os modelos devem tratar do aspecto de IHC de maneira separada, o desenvolvimento de IU baseado em modelos deve ser integrado ao restante da arquitetura da aplicação (desafio 9), para permitir desempenho e escalabilidade convenientes, considerando especialmente os requisitos de produtos comerciais [72, 38].

Algumas propostas para conciliar as disciplinas de IHC e Engenharia de Software no contexto do desenvolvimento baseado em modelos têm sido feitas [12, 23], mas as ferramentas existentes não são apropriadas para este propósito [37]. Esta lacuna torna-se ainda maior no contexto de SI quando a disciplina de Modelagem de Processo de Negócio é incluída. Assim, integrar a construção de IU ao restante de SI ainda é um desafio (desafio 10).

A utilização de padrões de interação juntamente com o desenvolvimento de IU baseado em modelos aumenta a efetividade e produtividade, já que a maioria das abordagens recentes de Engenharia de Software combinam padrões e modelos [21].

Métodos de IHC (Interação Humano- Computador) não têm sido suficientemente utilizados em todo o ciclo de vida do software [37]. Existe uma lacuna entre os conceitos e padrões de IHC e a geração automática de IU. É necessária uma descrição mais formal de padrões de IHC [60] para que sejam combinados com a transformação automatizada de modelos e geração de código de IU.

Por essas razões, e embora as ferramentas de desenvolvimento de IU baseadas em modelos sejam mais sofisticadas que as ferramentas de geração anteriores, elas não se tornaram amplamente populares na indústria de software [37]. Muitos desenvolvedores ainda usam construtores de leiaute, *toolkits* e linguagens de programação para construir IU para sistemas interativos.

Portanto, para compreender e solucionar os desafios envolvidos na construção de IU baseada em modelos é preciso entender que características de aparência e com-

portamento estão envolvidas na construção de IU e como representar estas características em um metamodelo conceitual. Também é necessário compreender como construir um componente que permita a geração automática de IU a partir de tal metamodelo, e em que momento do ciclo de vida de um software o componente irá atuar. Esses e outros assuntos pertinentes a geracao de IU sao tratados no presente trabalho, para um contexto particular: o de Sistemas de Informação (SI).

1.2 Objetivos do Trabalho

O principal objetivo deste trabalho é propor uma abordagem baseada em modelos para construção e evolução de IU para SI. Esta abordagem deve envolver um conjunto de metamodelos para construção de IU e uma arquitetura de um componente de software que auxilie na geração automatizada.

A abordagem proposta supera algumas das limitações das abordagens atuais, seguindo os princípios da Engenharia de IU baseada em modelos. Aspectos estruturais e tecnológicos de software, especialmente quanto à apresentação das informações e ao comportamento do SI em resposta à interação dos usuários são contemplados nesta abordagem.

1.3 Contribuições do Trabalho

Para alcançar os objetivos propostos, este trabalho contribui com uma compreensão de como produzir abordagens baseadas em modelos para desenvolvimento de IU. Neste sentido, a partir dos desafios identificados nesta área de pesquisa, foi gerado um conjunto de requisitos para abordagens de construção de IU baseada em modelos.

Foi proposta uma abordagem que atende a estes requisitos por meio de um conjunto de metamodelos para construção de IU e de uma arquitetura de componente de software baseada em estereótipos. O desenvolvimento da abordagem proposta originou soluções para alguns desafios relacionados à Engenharia de IU baseada em modelos, dentre os quais se destacam:

- A definição de um Metamodelo de IHC que contempla requisitos para a descrição de IU no contexto de SI, permitindo a decomposição dos componentes de IU e a descrição dos seus comportamentos [14, 25];
- O projeto de um componente de software que implementa uma solução para o problema da transformação entre modelo abstrato e componentes concretos de IU [15];

- A integração entre a arquitetura do componente de IU e os demais componentes de SI, relacionando esta abordagem com todo o processo de desenvolvimento de SI [25, 34].

Os resultados deste trabalho foram obtidos no contexto de uma macro-arquitetura de *framework* baseado em modelos para gestão de SI desenvolvida pelo Grupo de Pesquisa em Engenharia de Software e Banco de Dados do INF/UFG [2]. A integração do componente de gerência de IU, aqui descrito, com os demais componentes daquela arquitetura pode ser vista em [15, 19].

Durante o desenvolvimento do trabalho foram escritos artigos descrevendo resultados parciais deste projeto de pesquisa. Um artigo foi publicado no III Workshop de Teses e Dissertações em Sistemas de Informação [14], dois artigos foram publicados no Primeiro Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos [15, 34], um foi publicado no VIII Encontro Anual de Computação [25], e outro artigo foi aceito para publicação no VII Simpósio Brasileiro de Sistemas de Informação [19].

1.4 Metodologia de Desenvolvimento do Trabalho

O desenvolvimento desta pesquisa foi baseado em cinco grandes atividades: (1) Fundamentação Teórica; (2) Eliciação de Requisitos; (3) Design; (4) Avaliação; e (5) Documentação. As atividades 1 a 4 foram realizadas de forma sequencial, e a atividade 5 foi desenvolvida concorrentemente às demais atividades.

A Fundamentação Teórica envolveu uma revisão de literatura com a finalidade de investigar abordagens para geração automática de IU. Esta investigação foi fundamentada nas seguintes **questões de pesquisa**:

1. Quais requisitos devem ser atendidos em uma abordagem de construção de IU baseada em modelos?
2. Quais são as características de aparência e comportamento envolvidas na construção de IU para SI?
3. Como representar estas características em um modelo conceitual?
4. Como construir uma ferramenta que permita a geração automática de interfaces de usuário a partir de tal modelo?

A partir destas questões, foi realizada a revisão por meio da busca nas bases bibliográficas da ACM, IEEE e Scopus. Foram encontradas algumas abordagens para construção de IU baseada em modelos, que estão relatados na Seção 2.2.

A atividade de Eliciação de Requisitos identificou os requisitos necessários para a geração de IU no contexto de SI, definindo os objetivos da abordagem proposta

neste trabalho. Na busca por resposta à primeira questão de pesquisa, foi elaborada uma taxonomia de requisitos para abordagens de construção de IU baseada em modelos, apresentada na Seção 2.3. Com base nestes requisitos, foi realizada comparação entre as abordagens identificadas na revisão bibliográfica, discutida na Seção 2.5. Foram desenvolvidos metamodelos que trazem as características identificadas pela segunda e terceira questões de pesquisa apresentados no Capítulo 3.

A atividade de Design especificou o projeto arquitetural e detalhado de um componente de software que realiza a geração de IU para SI, contemplando os requisitos identificados na atividade anterior. Em resposta à quarta questão de pesquisa, no Capítulo 4 foi especificado um componente de software que gerencia a construção de IU com base na abordagem desenvolvida neste trabalho.

A Avaliação da abordagem foi efetuada através de uma prova de conceito envolvendo a demonstração da utilização do componente para a geração de IU em SI, apresentada no Capítulo 5.

1.5 Organização do Trabalho

Este capítulo apresentou os principais conceitos e discutiu os desafios encontrados em abordagens de construção baseada em modelos. Também foram apresentados os objetivos, contribuições e a metodologia de desenvolvimento deste trabalho.

O Capítulo 2 apresenta conceitos fundamentais relacionados à construção de IU e, com base nos desafios discutidos no Capítulo 1, define uma série de requisitos que devem ser atendidos nestas abordagens para que os desafios citados sejam superados. O capítulo também compara as abordagens atuais em relação aos requisitos expostos, identificando aqueles que não foram resolvidos.

O Capítulo 3 descreve os metamodelos desenvolvidos neste trabalho. O Metamodelo de Domínio descreve os conceitos do negócio. O Metamodelo de IHC introduz o conceito de Estereótipo de Interface e os aspectos de apresentação e de comportamento de IU considerados neste conceito. O capítulo também apresenta uma extensão do Metamodelo de Apresentação [17], proposto para trabalhar de forma conjunta com o Metamodelo de IHC na geração automatizada de IU.

O Capítulo 4 explica a arquitetura de software proposta para a construção automatizada de IU em SI. São exploradas diferentes visões arquiteturais complementares que sintetizam a abordagem proposta para geração de IU para SI.

O Capítulo 5 discute cenários que ilustram a utilização da arquitetura de software para gerência de IU. O capítulo discute a aplicação da abordagem proposta em termos da identificação de Estereótipos de IU para SI, e da utilização da arquitetura para construção de IU para SI baseada em Estereótipos de IU.

Finalmente, o Capítulo 6 apresenta as considerações finais deste trabalho de pesquisa. São discutidas as contribuições e limitações do trabalho, bem como uma lista de extensões que poderiam melhorar ou complementar os resultados aqui apresentados.

CAPÍTULO 2

Desenvolvimento de IU Baseado em Modelos

É por meio de Interfaces de Usuário (IU) que acontece a interação entre os envolvidos nos SI. A interação do usuário com a aplicação é definida como as ações que tomam lugar entre o ser humano e a interface, na qual esta atua como uma ligação para comunicação com a funcionalidade do sistema de software subjacente a fim de realizar uma tarefa particular [69]. Portanto, no processo de interação existem três atores principais: o sistema de software (aplicação), o usuário e a interface entre eles.

O desenvolvimento de IU compreende, minimamente, a modelagem de aspectos de apresentação e de comportamento da interface. A modelagem de apresentação corresponde à descrição da aparência das informações da aplicação, e de como elas são apresentadas para o usuário. Este aspecto depende diretamente das informações que serão manipuladas por meio da interface.

Já a modelagem de aspectos de comportamento consiste na descrição da forma pela qual a interface responde às interações do usuário com a aplicação. O comportamento da interface é influenciado pelo tipo de aplicação manipulado. Espera-se, por exemplo, que aplicações do mesmo tipo tenham comportamentos similares.

Este capítulo discute os avanços já obtidos e as limitações atuais das abordagens para geração de IU baseada em modelos. Para isso, são analisados: (i) os conceitos fundamentais da Engenharia de IU; (ii) os desafios para a construção de IU baseada em modelos; e (iii) os requisitos que devem ser encontrados em abordagens de construção de IU baseada em modelos para SI. O capítulo apresenta, ainda, uma comparação entre abordagens para construção de IU encontradas na literatura.

2.1 Engenharia de Interfaces de Usuário

A necessidade de automatizar a construção de IU e de ter uma metodologia para seu desenvolvimento durante todo o ciclo de vida de SI, considerando os tradicionais conceitos de Engenharia de Software, levou ao surgimento da Engenharia de Interfaces de Usuário [73]. Esta engenharia é uma disciplina que pesquisa e desenvolve soluções para os desafios emergentes na modelagem de IU baseada em modelos, tais como a complexidade

dos dispositivos e estilos de interação, a multiplicidade de ambientes de trabalho, a diversidade de contextos de uso, e a heterogeneidade de arquiteturas de software.

O conhecimento gerado por esta disciplina, situada na intersecção entre as disciplinas de Engenharia de Software, Interação Homem-Computador e Fatores Humanos, está voltado, principalmente, para a pesquisa de metodologias para desenvolver IU, considerando todo o ciclo de vida do software [73].

A Engenharia de IU busca o desenvolvimento de metodologias que contemplem uma série de modelos que representam as diferentes facetas de IU [74].

A abordagem baseada em modelos foi introduzida para apoiar a especificação e *design* de sistemas interativos em um nível semântico, conceitual e abstrato, como uma alternativa para lidar com questões de baixo nível de implementação no começo do ciclo de vida de desenvolvimento [1]. Os modelos se tornam artefatos "vivos" que servem como entrada para ferramentas de geração de código, diminuindo o esforço dos desenvolvedores.

No entanto, o emprego desta abordagem requer uma linguagem para especificar modelos, definindo transformações e descrevendo metamodelos. O uso de modelos permite aos desenvolvedores manter e desenvolver componentes complexos encapsulados em modelos e mapeamentos, simplificando, assim, o processo de desenvolvimento [4].

As bases da Engenharia de IU baseadas em modelos são estabelecidas pelo *framework* de referência CAMELEON (*Context Aware Modelling for Enabling and Leveraging Effective interactiON*) [9], que cobre tanto o tempo de *design* como o tempo de execução de IU multi-plataforma, provendo uma estrutura conceitual que apoia o desenvolvimento de IU para uma ampla variedade de dispositivos computacionais.

O *framework* de referência CAMELEON foi proposto como uma abordagem para desenvolvimento de IU multi-contexto, com uma estrutura conceitual que decompõe o desenvolvimento de IU em quatro passos, resumidos na Figura 2.1.

1. Modelagem de tarefas e conceitos: descreve as tarefas de usuário e os conceitos de domínio, da forma como são requeridos pelas tarefas a serem realizadas.
2. Modelagem de Interface de Usuário abstrata: define os componentes abstratos individuais e agrupadores, o esquema de navegação, e os objetos de interação abstrata.
3. Modelagem de Interface de Usuário concreta: realiza uma interface de usuário abstrata em um contexto de uso com base em objetos concretos de interação. Apesar de definir o leiaute dos *widgets* e a navegação da interface, o modelo concreto ainda é um protótipo da interface de usuário.
4. Construção da Interface de Usuário final: cria a interface de usuário, ou seja, aquela que é executada em um ambiente específico.

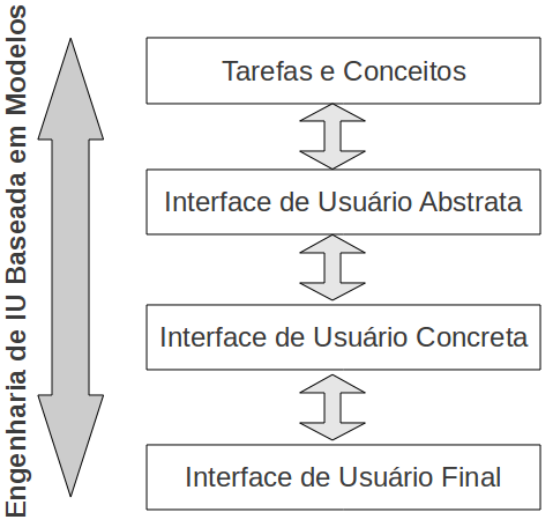


Figura 2.1: *Engenharia de IU Dirigida por Modelos (adaptada de [5])*

A Modelagem de Tarefas e Conceitos envolve a descrição das tarefas de usuário e os conceitos orientados ao domínio requeridos para a realização das tarefas. Para isto, é necessário produzir um modelo de tarefas e um modelo de domínio.

Para modelar o domínio é necessário descrever as classes de objetos manipuladas pelos usuários enquanto interagem com o sistema. Tipicamente são utilizados diagramas de classes da UML, modelo entidade-relacionamento ou um modelo orientado a objetos.

A modelagem de tarefas de usuário é uma técnica bem compreendida e amplamente aceita nas atividades do design de IU centrado no usuário [74, 50, 61]. Ela envolve a descrição das tarefas que o usuário deve desempenhar a fim de alcançar um objetivo quando interage com um sistema computacional, e o relacionamento entre estas tarefas. Tipicamente elas são decompostas em uma hierarquia de tarefas. Tarefas que são sujeitas a outras tarefas são consideradas tarefas dependentes, e aquelas que não tem uma conexão imediata com outras tarefas são tarefas independentes [64]. Muitos modelos de tarefas têm sido propostos na literatura e alguns dos principais modelos são descritos e comparados em [32].

Os demais passos – Modelagem da Interface de Usuário Abstrata, Modelagem da Interface de Usuário Concreta e Construção de Interface de Usuário Final – envolvem a modelagem de IU e os passos para a sua construção.

A abordagem da construção de IU baseada em modelos apresenta vantagens significativas com relação à abordagem de mapeamento manual de necessidades de IU para linguagens de programação. O uso de modelos de alto nível de abstração permite representar de forma mais clara os requisitos dos usuários com relação às IU. A facilidade em modelar interfaces por não especialistas motiva o uso de modelos, além de permitir que *design*ers sejam capazes de focar mais em aspectos de IHC e menos em detalhes de

implementação [4].

2.2 Abordagens para construção de IU baseada em modelos

Existem diferentes abordagens para modelagem de IU encontradas na literatura [59, 10, 16, 69, 76, 41]. Nos últimos anos muitos esforços têm sido realizados para desenvolver abordagens baseada em modelos para automatizar a construção de IU [54, 26, 42, 5, 22, 17, 36, 8, 20, 37, 43, 65, 31].

No entanto, a falta de consenso sobre os modelos mais adequados para descrever IU tem sido uma barreira para o uso intensivo de Engenharia de IU. Nenhum método tem realmente emergido como uma referência para a criação do software de IU [72, 1].

Essa falta de consenso pode ser explicada pela ausência de padrões universais de IHC, mas também porque as intenções e os objetivos das IU podem variar largamente de uma aplicação interativa para outra.

Existem vários métodos que instanciam um modelo de interação para representar IU para plataformas específicas. Como consequência, a migração para outras plataforma é uma tarefa difícil. USIXML [33] e UMLi [16] definem a interação de maneira abstrata. Porém, estas abordagens conseguem definir apenas IU genéricas, e não estão integradas em um processo de geração de código completo. Assim, elas deveriam especificar como a interface de usuário é apropriadamente conectada à funcionalidade do sistema [69].

Os desafios de modelagem de interação fomentaram a criação de uma multiplicidade de linguagens de padrões de IHC [70, 71, 23, 44, 7, 68]. No entanto, estas linguagens não são mutuamente compatíveis. Faltam descrições precisas dos elementos de interação e de como eles devem se comportar em uma implementação da solução [40].

Especialistas de IHC, do domínio da aplicação, e de engenharia de software deveriam poder expressar suas habilidades na forma de padrões, já que a comunicação em equipes interdisciplinares é um dos principais desafios dos praticantes de IHC [40].

A união dos conceitos de padrão de interação com os de descrição de IU, permite formar padrões concretos de interação [37]. A Biblioteca de Padrões para Design de Interação, apresentada em [70], identifica contextos de *design*, necessidades do usuário e da aplicação. Já em [44] são definidos quinze Padrões de *Layout* de Tela para RIA (*Rich Internet Applications*).

2.3 Requisitos para Construção de IU em SI

Uma descrição de IU em alto nível é realizada de maneira adequada com um metamodelo de IU que especifique as características essenciais para o projeto de IHC. Um metamodelo de IU (ou de IHC) deveria contemplar [45]:

- Um conjunto de termos precisamente definidos sobre IHC;
- Definições estruturadas da aplicação de conceitos de IHC para a criação de modelos;
- Alta expressividade, permitindo descrever cada aspecto da interação;
- Coerência e interoperabilidade do resultado da modelagem;
- Independência de tecnologias e de formas de armazenamento;
- Escalabilidade, oferecendo meios de definição de diferentes níveis de abstração.

Um aspecto importante para um metamodelo é a sua capacidade de incorporar *padrões de projeto*, ou seja, soluções provadas para problemas recorrentes. Padrões de projeto de interação focam na solução de problemas de usabilidade dos sistemas. Esses padrões se correlacionam e se complementam, formando um acervo de boas práticas de interação que podem ser aplicadas em diversos projetos de IU de forma consistente [45].

Com base nestas propriedades, e nas dificuldades citadas na Seção 1.1, é possível identificar um conjunto de requisitos para abordagens de construção de IU baseada em modelos para SI [72, 45, 35, 76, 38, 51, 60]. Estes requisitos foram organizados em uma taxonomia de Requisitos para Abordagens de Construção de IU Baseadas em Modelos, que é apresentada na Figura 2.2.

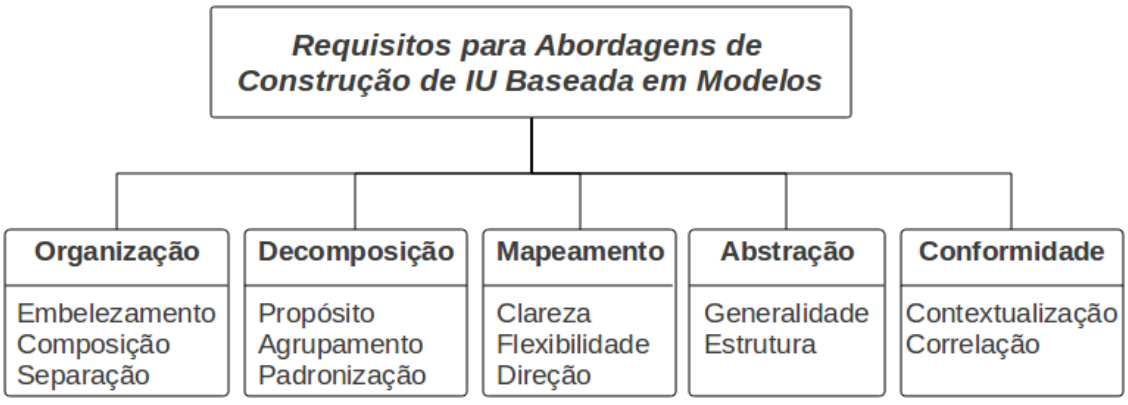


Figura 2.2: *Requisitos para Abordagens de Construção de IU baseada em Modelos*

Os requisitos foram divididos em cinco classes relacionadas ao desenvolvimento de interfaces baseado em modelos: Organização, Decomposição, Mapeamento, Abstração e Conformidade.

2.3.1 Requisitos de Organização

A classe de requisitos Organização agrupa os requisitos associados à organização da construção de IU, e suas características são Embelezamento, Composição e Separação.

O requisito de Embelezamento diz respeito ao apoio que a abordagem de construção de IU deve oferecer para que as IU geradas sejam embelezadas e refinadas ao longo do ciclo de vida.

O embelezamento envolve detalhes de aparência da interface, como o tipo e tamanho da fonte, cor de fundo, imagens de ícones, posicionamento dos componentes, entre outros detalhes relacionados à estética de IU.

A Composição diz respeito ao agrupamento de elementos no leiaute da interface. A disposição dos elementos na interface depende da importância de cada um dos elementos e de características de usabilidade.

A construção de IU envolve os conceitos do domínio da aplicação, porém estas preocupações devem ser consideradas de maneira separada. Esta Separação favorece a manutenção dos conceitos do negócio e das IU de forma isolada.

2.3.2 Requisitos de Decomposição

A classe de requisitos Decomposição reúne as seguintes características relacionadas à subdivisão dos componentes de IU: Propósito, Agrupamento e Padronização.

Existem IU com diferentes propósitos, e ter uma biblioteca onde estão identificados alguns dos Propósitos recorrentes de IU pode auxiliar na produtividade da construção de IU. Um dos propósitos mais utilizados em SI são as IU do tipo CRUD (*Create, Read, Update, Delete*), empregadas para a manipulação de informações do negócio.

O conceito de elemento dominante, aquele que faz o Agrupamento de elementos simples para a formação de elementos mais complexos, pode solucionar problemas recorrentes nas IU. Elementos isolados não tem um propósito para utilização, contudo, reunidos com um objetivo eles se tornam reutiliáveis em outros contextos semelhantes, favorecendo a produtividade na construção de IU e promovendo a Padronização da aparência e do comportamento das IU.

2.3.3 Requisitos de Mapeamento

A classe de requisitos Mapeamento diz respeito aos requisitos necessários para a transformação de modelos para a construção automatizada de IU. Clareza, Flexibilidade e Direção são características desta propriedade.

Para que os modelos sejam utilizados com eficiência, as regras de transformação devem ser especificadas de forma clara e precisa entre o metamodelo abstrato e sua

especificação concreta. Além disso, alguns componentes abstratos podem ter mais de um mapeamento para elementos concretos, e deve-se possibilitar a Flexibilidade de configurar este mapeamento.

A Direção é observada quando os mapeamentos são realizados da maneira adequada, de uma representação mais abstrata para uma mais concreta.

2.3.4 Requisitos de Abstração

A classe de requisitos Abstração apresenta as características esperadas para que uma abordagem seja independente de uma tecnologia específica.

O requisito de Generalidade está relacionado com a manutenção de um alto nível de abstração e generalidade nos modelos empregados para especificação de IU, o que torna esta especificação independente de tecnologia.

O requisito de Estrutura deve ser observado pela especificação estruturada de meta-características para construção de IU. Dessa forma, as abordagens para construção de IU deveriam oferecer um conjunto de termos precisamente definidos sobre IHC, abrangendo conceitos de apresentação e comportamento de IU.

2.3.5 Requisitos de Conformidade

A classe de requisitos Conformidade diz respeito a construir as IU de acordo com a necessidade das aplicações. Para isto, as características de Contextualização e Correlação são esperadas.

A Correlação envolve a associação entre as funcionalidades e conceitos do negócio com a especificação da interface, de modo que seja possível compreender quais conceitos e funcionalidades do negócio devem estar presentes na interface de usuário.

A Contextualização tem a preocupação de apresentar a interface de forma adequada ao domínio das informações do negócio, seus tipos e relacionamentos.

2.4 Correlação entre Desafios e Requisitos para abordagens de Construção de IU

Os desafios identificados na Seção 1.1 guiaram o desenvolvimento dos requisitos para abordagens de construção de IU baseada em modelos, apresentados na Seção 2.3.

O Desafio 1, relacionado a necessidade de ter uma linguagem estruturada para descrição de IU, está relacionado com o requisito Estrutura, da Classe Abstração, que prevê uma descrição estruturada das meta-características para construção de IU.

O Desafio 2, que diz respeito ao problema do mapeamento entre modelos, está relacionado com os requisitos da classe Mapeamento: Clareza, Flexibilidade e Direção. A presença destes requisitos possibilitam o mapeamento adequado na aplicação de abordagens de construção de IU baseada em modelos.

O Desafio 3 considera o embelezamento das IU, e ele é considerado pelo requisito de Embelezamento da classe Organização. O Desafio 4, que corresponde a especificação do leiaute, é relacionado com o requisito de Composição da classe Organização, que preocupa-se com a disposição dos elementos nas IU.

O Desafio 5, relacionado as informações do domínio da aplicação consideradas para a construção de IU, está relacionado pelo requisito de Contextualização da classe Conformidade, que prevê que estas informações de domínio devem ser consideradas de forma separada na construção de IU.

O Desafio 6, que corresponde ao apoio à decomposição de IU, é considerado pela classe Decomposição, composta pelos requisitos de Propósito, Agrupamento e Padronização. O Desafio 7 considera a separação de preocupações, e está relacionado com o requisito Separação da classe Organização.

O Desafio 8 compreende a correlação entre os modelos envolvidos na abordagem. Este desafio é relacionado ao requisito de Correlação da classe Conformidade. A integração entre os demais aspectos envolvidos na construção de SI, que é o Desafio 9, está relacionado com a classe de Conformidade.

O Desafio 10, que corresponde a integração de padrões de IHC e o desenvolvimento de IU baseado em modelos, é considerado pelo requisito de Generalidade da classe Abstração. Deste modo, para que padrões de IHC sejam agregados ao desenvolvimento de IU baseado em modelos, é necessário considerar os modelos independentes de tecnologia com os padrões de IHC, que também são abstraídos de forma genérica. Esta agregação possibilita reutilização de padrões e qualidade nas IU geradas automaticamente.

2.5 Comparação entre abordagens

Na busca por abordagens para construção de IU baseada em modelos recentes (2007 à 2010), que possuem maior subconjunto de requisitos para o contexto de SI, foram selecionados alguns trabalhos relevantes, baseando-se nos seguintes critérios: ser uma abordagem baseada em modelos, ser aplicável a IU WIMP (*windows, icons, menus and pointers*), e ter uma implementação da proposta. A seguir, são analisadas as abordagens de construção de IU, de forma semelhante ao trabalho de [35].

A Tabela 2.1 apresenta a comparação das abordagens em relação aos requisitos apresentados na Seção 2.3. Os campos preenchidos com "++" indicam que o critério em questão foi totalmente atendido na abordagem, isto é, houve preocupação em exibir

o requisito na abordagem. Os campos preenchidos com "+" apontam que o critério em discussão foi atendido de forma parcial, ou seja, algumas partes do requisito são atendidas. Já os campos preenchidos com "-" indicam que o critério não foi atendido pela abordagem, isto é, o requisito é ausente na abordagem, ou na descrição dela.

Tabela 2.1: *Comparação entre as Abordagens para Construção de IU Baseada em Modelos*

Requisitos para Abordagens de Construção de IU	*Malai [5]*	*Metamodelo Conceitual de Discurso [22, 55]*	*GDIG [17]*	*Modelo Abstrato de Interação [69]*	*Metamodelo para Aplicações Web [8]*	*Abordagem baseada em IMML [13]*
Embelezamento (Organização)	-	-	-	-	+	-
Composição (Organização)	+	-	+	-	++	+
Separação (Organização)	+	-	+	-	-	++
Propósito (Decomposição)	++	++	+	++	+	++
Agrupamento (Decomposição)	-	-	++	+	++	++
Padronização (Decomposição)	++	+	+	+	++	++
Clareza (Mapeamento)	++	-	-	-	++	++
Flexibilidade (Mapeamento)	-	-	+	+	+	+
Direção (Mapeamento)	-	-	-	-	+	++
Generalidade (Abstração)	++	++	+	++	++	++
Estrutura (Abstração)	-	++	++	-	++	++
Contextualização (Conformidade)	+	-	++	+	+	+
Correlação (Conformidade)	-	-	+	-	+	++
Legenda: -: Não atendido; **+**: Parcialmente Atendido; **++**: Totalmente atendido						

Entre os metamodelos citados, se destacam o Metamodelo de Aplicações *Web* e a Abordagem baseada em IMML, por atenderem a um maior número de requisitos. O Malai [5] é o mais aderente ao CAMELEON. No Metamodelo Conceitual de Discurso [22, 6, 55] nota-se maior poder de expressão em termos de interação, visto que o uso de atos comunicativos torna esta iniciativa mais próxima da comunicação humana.

O Modelo Abstrato de Interação [69] é o mais abrangente em termos de definição de taxonomia para as formas de interação. Já o Metamodelo de Interface [17] é mais restrito em seu objetivo, mas apresenta uma ferramenta para geração automática de IU. As interfaces geradas são integradas, embora não acopladas, ao restante do código de SI. O Modelo de Domínio e de Interação [13] relaciona a lógica de negócio à interface, e mostra a aplicação de regras de transformação para a geração de IU.

Como mostra a Tabela 2.1, Malai e o Modelo Abstrato de Interação não fornecem uma definição estruturada dos conceitos, dificultando a compreensão do relacionamento entre os padrões de interação e da forma como estes são aplicados, pois o metamodelo é descrito apenas textualmente.

O conceito de elemento dominante, sendo um auxílio para a organização dos elementos na interface bem como dos comportamentos esperados, não é previsto no Metamodelo Conceitual de Discurso, nem no Malai. O Modelo Abstrato de Interação utiliza parcialmente este conceito, já que ele é empregado apenas para o aspecto de apresentação da interface. Na abordagem baseada em IMML, o conceito de elemento dominante aparece apenas em nível concreto, onde são produzidos templates para a produção do código final.

Além disso, não há preocupação com a separação da descrição dos elementos de IU com os conceitos da lógica de negócio no Modelo Abstrato de Interação, no Metamodelo Conceitual de Discurso e no Metamodelo de Aplicação Web. Eles também não correlacionam a modelagem da interface com a lógica de negócio e preocupam-se parcialmente com o formato e tipos de dados do negócio. Isto torna o desenvolvimento de IU isolado do restante do desenvolvimento da aplicação.

A preocupação com o refinamento da interface, bem como com aspectos de estilo, são propriedades que são deixadas em segundo plano, ou ainda ignoradas pelos metamodelos aqui discutidos. Não é fácil alterar o comportamento e a aparência da interface modelada, mas isto faz com que o processo de manutenção seja caro e propenso a erros.

Com o Metamodelo de Interface é possível gerar apenas um tipo de interface (CRUD). Também no Metamodelo de Aplicação Web podem ser construídos apenas alguns estereótipos: formulário, pesquisa, lista e mestre-detalhe.

O Metamodelo de Discurso traz conceitos interessantes de comunicação, mas não é o suficiente para expressar como o sistema computacional irá se comportar após a interação com o usuário. A abordagem baseada em IMML é interessante nesse sentido, pois define um conjunto de estruturas de interação, que organizam as ações executadas pelo usuário.

O Metamodelo de Aplicação Web é o único que não auxilia a especificação do comportamento. Os demais metamodelos, com exceção do Metamodelo de Interface, são

baseados em máquina de estados finita para a especificação dos aspectos comportamentais da interface.

Apesar de existirem muitas iniciativas para construção de IU baseada em modelos, não há consenso sobre qual metamodelo melhor representa as características de IU. Neste capítulo foram extraídas características de qualidade para metamodelos de IU no contexto de SI, reunindo primitivas de modelagem apresentadas em diferentes trabalhos.

Após a análise da Tabela 2.1, conclui-se que os trabalhos existentes não atendem por completo as características de qualidade identificadas. Esta avaliação motivou a definição de uma nova abordagem com um conjunto de metamodelos que busquem atender a estes requisitos de qualidade. O conjunto de metamodelos produzidos para esta abordagem é apresentado no próximo capítulo.

CAPÍTULO 3

Metamodelos para IU

Um metamodelo define uma linguagem de modelagem para escrever modelos de determinado domínio. Os três metamodelos apresentados neste capítudo foram projetados para atender aos requisitos para construção de IU em SI apresentados no Capítulo 2.

O Metamodelo de Domínio (Seção 3.1) descreve os dados do negócio e as regras associadas, que serão aplicadas pelo Sistema de Informação.

O conceito de Estereótipo de Interface, introduzido no Metamodelo de IHC (Seção 3.2), possibilita a modelagem de IU em um nível abstrato, facilitando sua reutilização em uma ampla gama de SI.

Já o Metamodelo de Apresentação (Seção 3.3) permite a descrição de IU em um nível concreto. Ele estende a proposta de [17], trabalhando em conjunto com o Metamodelo de IHC na geração automatizada de IU.

3.1 Metamodelo de Domínio

Um Sistema de Informação pode ser descrito por um conjunto de metadados de contexto de negócio. O Metamodelo de Domínio de Negócio, proposto em [15], representa esses metadados, possibilitando a descrição dos conceitos do negócio. O Metamodelo apresentado na Figura 3.1 é uma extensão do Modelo de Meta-Objetos Complexos [17], que por sua vez, é uma extensão do Modelo Entidade-Relacionamento.

Os principais construtores do Metamodelo de Domínio definem tipos específicos de **Elementos de Negócio**: **Entidade**, **Entidade Fraca**, **Consulta** e **Relacionamento**.

Uma Entidade descreve um conceito de negócio de um Sistema de Informação. Uma Entidade Fraca é um conceito do negócio que depende de um relacionamento com outros conceitos para ser identificado.

Um Relacionamento representa a associação de dois elementos de negócio, independente de sua natureza; por exemplo, é possível associar dois Relacionamentos, distintos ou não. Dentro de um Relacionamento, cada elemento de negócio associado deve ter um Papel.

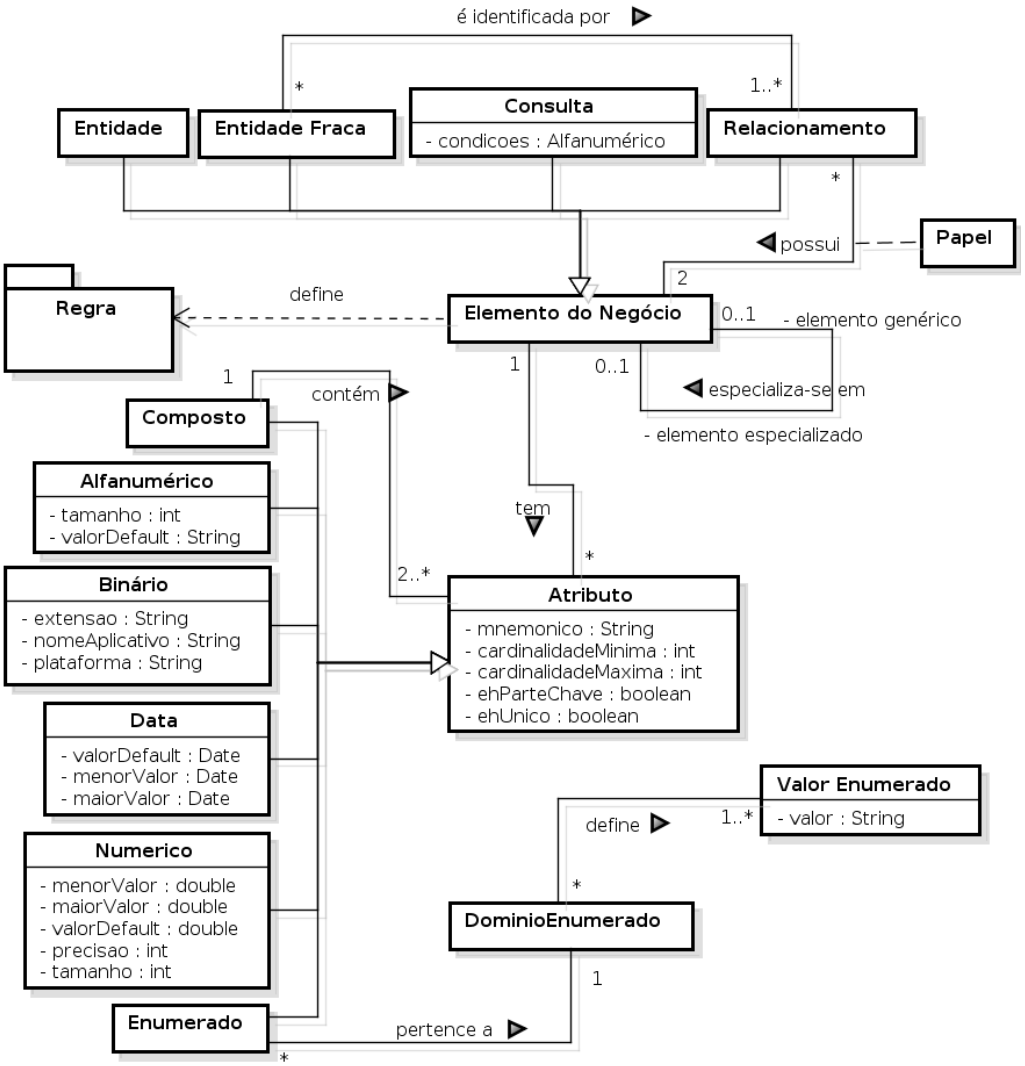


Figura 3.1: *Metamodelo de Domínio (adaptado de [15])*

Uma Consulta representa uma visão de uma combinação de determinados elementos de negócio, com um subconjunto de seus atributos, sendo equivalente ao conceito de Visão do modelo Relacional de dados. Este tipo de construtor permite a criação e o acesso a dados derivados no contexto de SI.

Elementos do Negócio podem especializar-se em outros elementos. Por exemplo, uma entidade pode se especializar em outra entidade, conforme o conceito de Herança da Orientação a Objetos. A especialização de elementos permite a identificação de subconjuntos de elementos que são de interesse para os SI.

Regras de Negócio são definidas sobre um Elemento do Negócio. Existem mecanismos específicos que abordam a construção destas regras, que podem referenciar diversos Elementos do Negócio a partir de um elemento base sobre o qual a regra é definida [2].

Todo Elemento do Negócio possui um número arbitrário de **Atributos**, os quais possuem as seguintes propriedades: mnemônico, que identifica o atributo; cardinalidade

mínima e máxima, que representa, respectivamente, o número mínimo e máximo de valores que o atributo pode ter; ehParteChave, que indica se o atributo faz parte da composição de algum atributo identificador do Elemento de Negócio; ehUnico, que determina se o atributo é um identificador, ou seja, não pode haver dois valores iguais para este atributo no mesmo Elemento de Negócio.

O Metamodelo de Domínio define uma taxonomia para os Atributos de Elementos do Negócio:

- **Composto**, representa uma agregação de atributos.
- **Alfanumérico**, representa informações textuais. Tem como propriedade o tamanho máximo do texto e um valor *default* (opcional).
- **Binário**, representa informações de arquivos binários. Tem como propriedades extensão (tipo do arquivo), aplicativo (indica qual aplicativo tem a preferência para abrir o binário) e plataforma (determina a plataforma necessária para abrir o binário).
- **Data**, representa dados temporais. Tem como propriedades valor *default* opcional, menor valor e maior valor.
- **Numérico**, representa informações numéricas. Tem como propriedade: menor valor, maior valor, valor *default*, precisão e tamanho máximo. Através da propriedade precisão pode-se identificar se o atributo numérico é inteiro (precisão igual a 0) ou real (precisão maior que 0).
- **Enumerado**, representa informações que assumem valores discretos. Um atributo Enumerado deve estar associado a um **Domínio Enumerado**, que define o conjunto de **Valores Enumerados**. A utilização destes conceitos facilita a reutilização de domínios enumerados por todas as aplicações de SI. Por exemplo, pode existir mais de um atributo que empregue o mesmo domínio enumerado. Os valores enumerados devem ter a propriedade valor, que armazena o conteúdo do valor enumerado.

Para exemplificar o uso do Metamodelo de Domínio, considere o Elemento de Negócio “Pessoa”, presente em inúmeros SI. Esse elemento pode ser representado como uma Entidade Pessoa, conforme mostra a Figura 3.2. Neste exemplo, a entidade possui os seguintes atributos (com seus respectivos domínios): Nome (alfanumérico), CPF (numérico), Data de Nascimento (data), Idade (numérico), Naturalidade (composto) que é composto por Estado (enumerado) e Cidade (enumerado).

A regra Valida Pessoa é definida sobre a entidade pessoa, sendo uma regra que valida os atributos da entidade Pessoa. Por exemplo, um dos itens verificados por esta regra poderia ser a consistência entre a cidade e o respectivo estado no atributo composto Naturalidade.

O Metamodelo de Domínio representa a descrição dos conceitos do negócio que serão maninpulados no Sistema de Informação através da interface de usuário. Porém, é

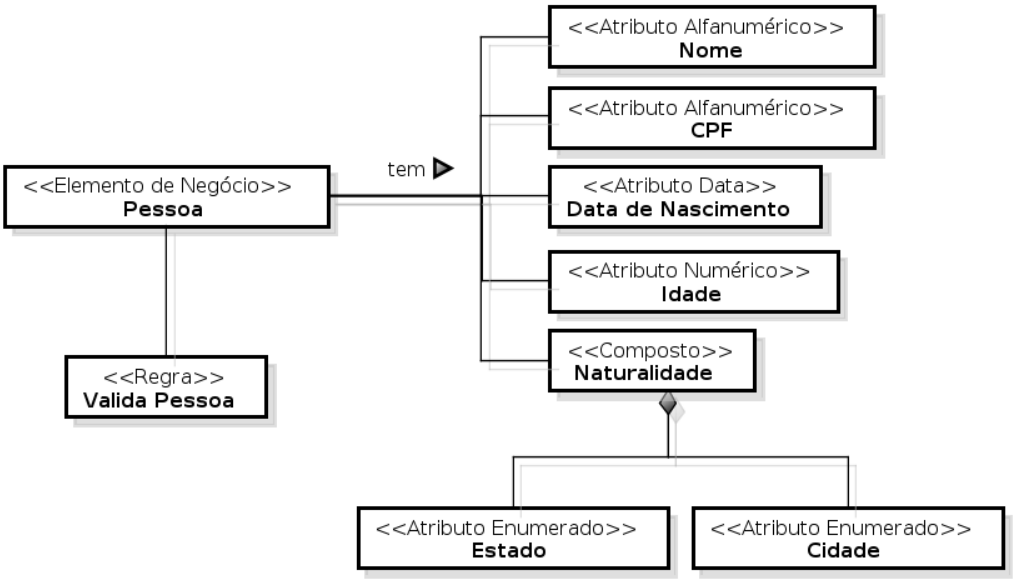


Figura 3.2: *Exemplo de Aplicação do Metamodelo de Domínio*

necessário considerar como estas informações serão apresentadas na interface, e qual a reação da interface para cada ação do usuário. Dete modo, a próxima seção apresenta um metamodelo que considera estas questões para a representação dos conceitos de negócio em IU de SI.

3.2 Metamodelo de IHC

O objetivo do Metamodelo de IHC é descrever as características de apresentação e comportamento de IU para SI em nível abstrato. Estas IU são baseadas em ícones, menus, ponteiros e janelas (WIMP, do inglês *Windows, Icons, Menus and Pointers*), fazem uso intensivo de informações estruturadas armazenadas em bancos de dados, e organizam as tarefas relacionadas aos processos de negócio de forma que façam sentido para os usuários [63].

O escopo de aplicação desse metamodelo abrange um vasto conjunto de aplicações de SI. No entanto, este escopo é restrito o suficiente para ser adequadamente representado por um único Metamodelo de IHC, descrevendo de forma integrada os aspectos de aparência e comportamento das IU.

Estes dois aspectos são tradados por componentes separados dentro do metamodelo, no intuito de isolar as propriedades de cada aspecto, mas são integrados como um único metamodelo para que seja explícita a maneira pela qual ambos os aspectos se correlacionam, representando IU complexas no âmbito de SI.

O metamodelo proposto especifica IU em alto nível de abstração e generalidade, integrando conceitos e padrões já consolidados pela comunidade de IHC, a fim de

proporcionar a geração automática do código de IU. A Figura 3.3 exibe a estrutura do Metamodelo de IHC para especificação de IU. O pacote Apresentação representa os aspectos de aparência de IU; no pacote Comportamento estão os metadados que permitem representar as ações em IU. O pacote Estereótipo de Interface reúne os metadados envolvidos para integrar este conceito com a apresentação e comportamento de IU.

O conceito-chave do Metamodelo de IHC é o de **Estereótipo de Interface**, ou simplesmente Estereótipo, que consiste em uma abstração da intenção da interface, independente da aplicação ou do Sistema de Informação subjacente. Assim, cada estereótipo é um **elemento dominante** de interface, organizando a apresentação dos elementos subordinados e seus respectivos comportamentos.

A definição estrutural da composição dos estereótipos está baseada em uma taxonomia para elementos abstratos de interface de usuário. Essa taxonomia corresponde ao **Aspecto de Apresentação** do metamodelo.

O comportamento dos estereótipos é definido com base em um mecanismo para especificação de ações associadas a eventos de interface, representando o **Aspecto de Comportamento** do metamodelo.

A ligação entre o comportamento e os elementos de interface, que formam um estereótipo, é representada através das correlações entre eventos capturados pelos elementos de apresentação e ações que descrevem o comportamento de reação ao evento.

A taxonomia de elementos de interface permite a especificação conceitual de IU, independentemente dos objetos reais de interação. Para cada elemento dessa taxonomia são definidos padrões de interação típicos, baseados na literatura de IHC [69, 55, 70, 44]. Assim, cada elemento de interface tem um comportamento esperado, que pode ser estendido a partir da associação do elemento a cada estereótipo de interface.

3.2.1 Aspecto de Apresentação de IHC

Para cada interface, determinado conceito de negócio de SI deve ser apresentado com um propósito específico. Um **Elemento de Interação** (ou Elemento de IHC), é um construtor que descreve o tipo de elemento de interface utilizado para a comunicação entre o sistema e o usuário.

A taxonomia definida para os Elementos de Interação possibilita a reutilização desses elementos entre diversas aplicações de SI, uma vez que estes elementos são uma definição genérica de que tipo de comunicação pode ser realizada na interface.

Um Elemento de Interação do tipo **Agrupador** representa uma composição de elementos de interação. Esses elementos complexos auxiliam na organização estrutural da interface, bem como no agrupamento de informações relacionadas, a fim de que o usuário possa perceber melhor o contexto da informação.

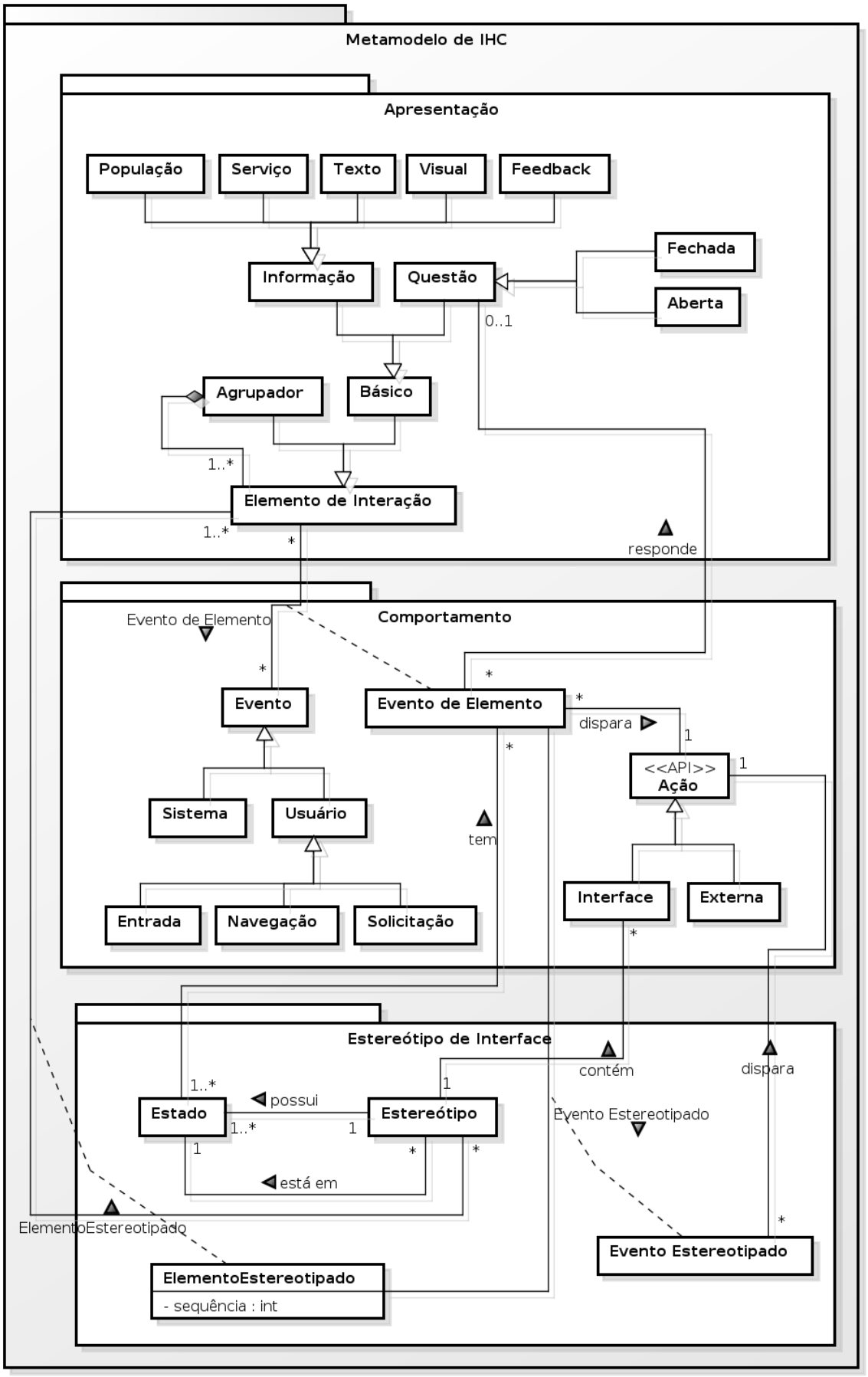


Figura 3.3: *Metamodelo de IHC*

Um Elemento de Interação do tipo **Básico** encapsula uma unidade essencial de comunicação do sistema com o usuário, da mesma forma que ocorre no conceito de atos comunicativos [22]. Elementos Básicos se dividem em dois tipos: **Questão** e **Informação**.

Um componente básico do tipo **Informação** é um elemento que deve apenas apresentar conteúdo informativo ao usuário. Conforme conceitos do Modelo Abstrato de Interação [69], as informações podem ser classificadas como **Texto**, apenas para fins de conhecimento do usuário; **População**, consistindo em um conjunto de instâncias de determinado conceito do negócio; **Serviço**, que executa uma funcionalidade; **Visual**,que representa imagens ou vídeos; ou **Feedback**, apresentando a resposta do sistema a uma interação do usuário.

Por exemplo, um elemento de negócio Notícia é uma informação do tipo Texto. Um conjunto de Notícias forma uma informação do tipo População. Um botão com o rótulo "Salvar" é uma informação do tipo serviço, pois comunica ao usuário que interagir com este elemento conduzirá o sistema a executar a ação Salvar. Uma figura ou foto exibida na interface é um tipo de informação visual. Um exemplo de informação do tipo *feedback* é uma caixa de alerta exibida ao usuário, notificando uma resposta do sistema à alguma ação do usuário.

Uma Questão é um elemento que comunica ao usuário um questionamento que requer uma resposta (interação) do usuário, que pode ser uma edição ou uma inserção de dados. Assim, um elemento questão deve conter uma identificação dos dados a serem manipulados pelo usuário (rótulo) e um espaço para manipulação dos valores dos dados, definido conforme o domínio da informação especificado no contexto de negócio. A resposta deve pertencer a este mesmo domínio de informação, que pode ser, conforme a taxonomia definida em [69]: enumerado, alfanumérico, numérico, data ou binário.

Uma Questão pode ser de dois tipos: **Aberta** ou **Fechada**. Uma questão pertencente ao domínio Enumerado é uma **Questão Fechada**, pela qual o usuário escolherá uma resposta dentre os dados desse domínio apresentados na interface. Questões que pertencem aos demais domínios (alfanumérico, numérico, data e binário) são do tipo **Aberta**.

Ambos os tipos de questão necessitam da informação de cardinalidade, presente no Metamodelo de Domínio, para que seja disponibilizado espaço adequado para o usuário responder à questão na interface.

A cardinalidade de uma questão é um par de números inteiros que define a quantidade mínima e máxima de dados que o usuário deve fornecer. Se a cardinalidade mínima for igual a zero, então a resposta da questão não é obrigatória. Se a cardinalidade máxima for maior que um, então é necessário um elemento do tipo agrupador que organize o espaço adequado para que seja permitido ao usuário entrar com a quantidade de informação necessária.

Os componentes específicos para a interface que representarão cada tipo de

questão devem ser definidos em nível concreto, no Metamodelo de Apresentação, conforme discute a Seção 3.3.

3.2.2 Aspecto de Comportamento de IHC

Dentre as diferentes formas de descrever a interação do usuário, os modelos baseados em **Máquina de Estados** são muito empregados, e podem ser derivados do modelo de discurso [28]. O Metamodelo de IHC contém a especificação de uma máquina de estados finita para descrever a interação, ou seja, a forma de reação da interface em resposta à interação do usuário. Por meio do conceito de Elemento de IHC, pode ser enumerado um conjunto de eventos de elemento para o Estereótipo. Cada um deles irá disparar uma ação.

Um evento consiste em uma ação que ocorre na interface em determinado elemento de interação, podendo provir tanto do usuário como do sistema. Estes **Eventos de Elemento**, uma vez ocorridos, devem disparar a execução de uma **Ação**.

Dentre os possíveis **Eventos de Usuário** existem eventos de **Entrada de Dados**, relacionados aos elementos do tipo questão do aspecto de apresentação; de **Navegação**, que são eventos relacionados a elementos de serviço que possibilitam alterar o estado da interface; e de **Solicitação**, que são eventos relacionados a elementos de serviço que acionam ações externas.

A ocorrência de um evento de elemento no Estereótipo caracteriza uma interação. As interações ocorrem em um **Estado**, que consiste em um conjunto de propriedades do sistema perceptíveis para o usuário. Conforme o **Estado** em que a interface se encontra, haverá um conjunto de **Eventos** que são de interesse do Estereótipo.

Um estado deve ter uma identificação e, dentro de um Estereótipo, um único estado deve ser identificado como estado inicial da interface. O fato de o Estereótipo ter estados e, durante sua execução estar em um estado, permite que a interface seja *quasi* modal, para prevenir o erro do usuário. *Quasi* modal é um estado no qual o usuário precisa fazer alguma ação física constante, a fim de manter o sistema naquele estado, de modo que não se esqueça que a interface está naquele modo. Um exemplo é o uso da tecla *shift* do teclado [57].

O metamodelo permite definir um conjunto de **Eventos de Elemento**, que são predefinidos para cada Elemento de Interação.

Eventos podem ocasionar a transição de um estado para outro. Para cada estado deve haver um conjunto de comportamentos em resposta aos eventos de usuário. Dada a ocorrência de um evento, é verificado se o evento é de interesse no estado corrente. Caso seja, é acionado um comportamento através da ação habilitada para aquela interação no estado corrente.

Dada a ocorrência de um evento em certo estado, seu comportamento é executado via **Ação**. Para cada evento de elemento (ou evento estereotipado, apresentado na Seção 3.2.3) em um estado, deve haver uma ação correspondente. Uma Ação consiste em um conjunto de comandos, restringidos por condições, que devem ser executados a partir do momento em que um evento é observado.

As ações caracterizam o comportamento executado após uma interação do usuário. Elas podem ser de **Interface** ou **Externa**. A primeira é executada pela interface, uma vez que sua execução não necessita do conhecimento da aplicação ou de outro sistema externo. Dois tipos notáveis de ação de interface são as ações de navegação, que realizam a transição entre interfaces, e a ações de transição de estados, que possibilitam a passagem de um estado para outro em uma mesma interface.

As ações externas são aquelas implementadas pela aplicação subjacente ou por outro sistema externo, mas são especificadas na modelagem de IU. Como as ações externas são especificadas de forma isolada do comportamento de interface, estas podem ser reutilizadas sempre que necessário. Por exemplo, uma ação que possibilita a consulta de um determinado tipo de informação do negócio pode ser utilizada em vários estereótipos.

3.2.3 Estereótipo de Interface

Em geral, os SI possuem funcionalidades e necessidades comuns de apresentação de informação. O conceito de **Estereótipo de Interface** permite capturar essas similaridades dos SI no que diz respeito a IU. Um Estereótipo modela uma forma de apresentação associada a um comportamento de interface recorrentes em diferentes aplicações de SI [14].

Este conceito integra e estende os conceitos de Contexto de Design [70] e de Padrões de *Layout* de Tela [44]. O conceito de contexto de *design* está relacionado com o propósito da interface, ou ainda o tipo de aplicação desenvolvida. Os contextos de *design* apresentam alguns tipos de aplicação para determinados domínios, separados em três categorias: Tipos de Sites, Experiência do Usuário e Tipos de Página. Por exemplo, portal e aplicação *Web* são tipos de site identificados. Gerenciador de Informações e Assistência (suporte) são do tipo Experiência do Usuário. *Blog*, Artigo e Contato são Tipos de Página identificados.

Já os padrões de *layout* de tela [44] abstraem alguns formatos de apresentação de IU, independente do domínio a que se aplicam. São identificados 15 padrões de tela, dentre eles Mestre-detalhe, que permite ao usuário permanecer na mesma tela enquanto navega entre os itens; Planilha, que oferece fácil edição, adição, visualização e totalização; Formulário e Portal. Do mesmo modo, um estereótipo deve abstrair a aparência de IU juntamente com seu comportamento.

Por meio do Estereótipo de Interface é possível abstrair as tarefas e interações com o usuário, bem como a forma de apresentação das informações para uma determinada tarefa. Cada Estereótipo deve determinar a forma de apresentação das informações para determinada necessidade de SI, bem como o comportamento da interface a cada interação com o usuário.

Por exemplo, formulários possibilitam a entrada de dados na aplicação pelo usuário; planilhas apresentam um conjunto de registros de determinado elemento de negócio; portais *Web* apresentam uma ampla variedade de informações organizadas em subsites, que podem conter funcionalidades específicas.

Um Estereótipo de Interface deve ter uma identificação única e conter elementos abstratos de interação. O estereótipo organiza esses elementos em relação às propriedades de apresentação e comportamento. Como um estereótipo pode ser reutilizado como um elemento de outro estereótipo, é possível definir elementos complexos de IHC. A partir de Estereótipos simples podem ser gerados tipos de Estereótipos mais complexos e importantes para SI, tais como páginas de consulta, planilhas, portais Web, e relatórios mestre-detalhe.

Um Estereótipo descreve IU para aplicações típicas de SI. Nele são definidos elementos-chave, comuns a todas as instâncias do estereótipo, deixando-se em aberto os elementos específicos de cada instância. Esses elementos são preenchidos conforme os conceitos de negócio da aplicação selecionada.

Um Estereótipo de Interface deve conter pelo menos um Elemento de Interação. Assim, um Elemento de Interação presente em determinado Estereótipo caracteriza um **Elemento Estereotipado**. Este elemento deve possuir um número de sequência, possibilitando que o Estereótipo organize seus elementos conforme a ordem especificada.

Os comportamentos do estereótipo devem ser definidos através da especificação de ações. Uma Ação pode envolver apenas a interface, como transição de estados, navegação e *feedback* ao usuário, não necessitando da intervenção da aplicação para ser executada. Logo, este tipo de comportamento é especificado como ação de interface.

Uma vez que um Elemento de Interação é associado a um estereótipo, este elemento pode estender os eventos de elemento com **Eventos Estereotipados**, possibilitando associar outra ação que será executada após a ocorrência do evento.

As ações de interface são predefinidas pelo estereótipo, já que estas não necessitam do conhecimento da aplicação para sua execução. Logo, todas as intâncias do estereótipo executarão o mesmo comportamento para ações de interface.

Já as ações que dependem da aplicação, ou de outros sistemas, são especificadas pelo estereótipo como ações externas. O mecanismo para especificação de ações é explicado na Seção 3.2.2.

O emprego do conceito de Estereótipo para a definição de IU promove a reuti-

lização. A abstração da apresentação e do comportamento de interfaces em estereótipos permite que os elementos abstraídos sejam instanciados em diferentes aplicações de diversos SI. Isto eleva a produtividade, já que a instanciação de estereótipos faz com que as interfaces sejam produzidas mais rapidamente por reaproveitar a aparência e comportamento de IU com os mesmos propósitos.

Além disso, tendo-se abstraído em um estereótipo a apresentação e o comportamento de uma classe de IU, promove-se a padronização e consistência, uma vez que a instanciação de qualquer interface irá gerar a mesma apresentação e o mesmo comportamento, com os mesmos propósitos definidos pelo estereótipo.

Um exemplo simples de aplicação do conceito de Estereótipo de Interface é o Formulário, um tipo de interface muito utilizado em SI. O Estereótipo Formulário tem o objetivo de receber informações do usuário para que a aplicação possa processá-las e armazená-las.

A Figura 3.4 apresenta uma abstração da forma de apresentação de um Formulário. O elemento Título é um agrupador que apresenta o elemento de negócio no qual se aplica o Formulário. Este agrupador contém outro agrupador, que é composto por elementos de interação do tipo Questão, já que o objetivo do estereótipo é a entrada de dados por parte do usuário.

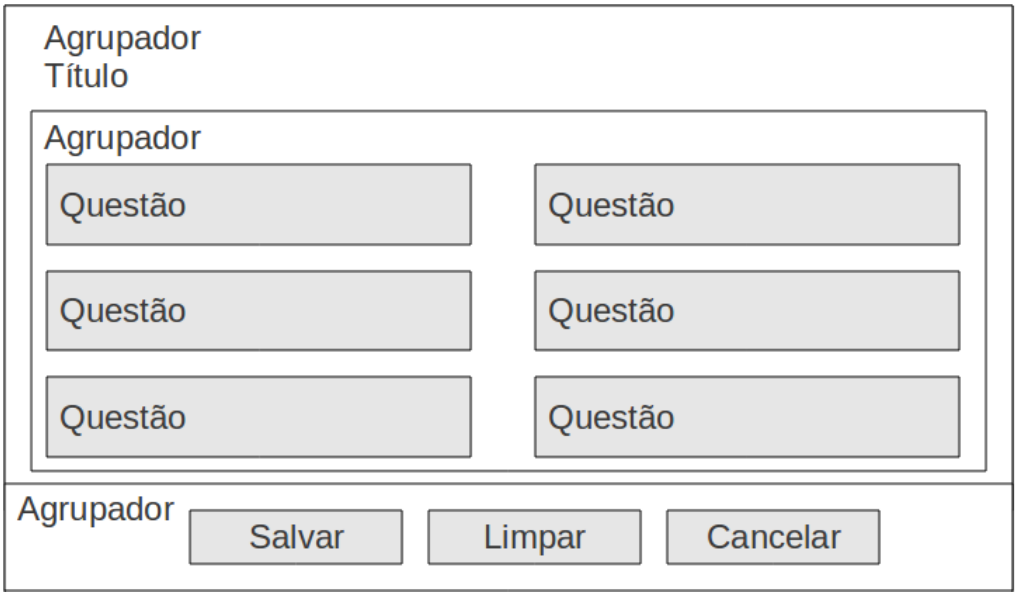


Figura 3.4: *Exemplo de Estereótipo - Formulário*

O número de questões a serem colocadas no formulário irá variar conforme o elemento de negócio a ser representado na interface, ou seja, para cada elemento de negócio haverá um número diferente de questões no formulário. Também faz parte do estereótipo Formulário um agrupador com elementos de interação do tipo Serviço, que acionarão as regras de processamento, implementadas pela aplicação subjacente, para as ações predefinidas pelo estereótipo.

O Estereótipo define o comportamento especificado por meio dessas ações. Para o estereótipo Formulário, as ações necessárias são: Salvar, Limpar e Cancelar. A ação Salvar deve implementar uma função que armazene as informações fornecidas pelo usuário. Essas informações são encapsuladas em um Elemento de Negócio representado no estereótipo.

A ação Limpar deve apagar todas as informações que o usuário colocou no formulário. A ação Cancelar fará com que seja fechado o formulário sem salvar qualquer informação. Como as ações Limpar e Cancelar independem da aplicação, elas são predefinidas pelo próprio estereótipo, sendo especificadas como ações de interface.

3.2.4 Exemplo de Utilização do Metamodelo de IHC

Um exemplo de utilização do Metamodelo de IHC para a modelagem de IU é apresentado na Figura 3.5. Neste exemplo, o elemento de negócio (entidade Pessoa), apresentada na Seção 3.1, foi o alvo da modelagem de IHC com base no estereótipo de Formulário.

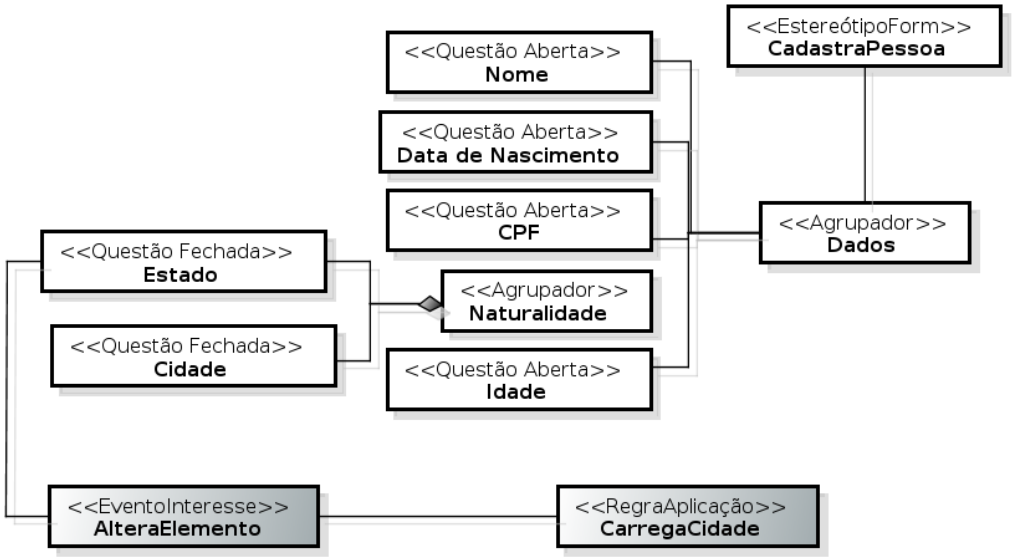


Figura 3.5: *Exemplo de Aplicação do Metamodelo de IHC*

O estereótipo Formulário associa os elementos de interação do tipo questão aos atributos do elemento de negócio (entidade Pessoa). O mapeamento realizado entre o Metamodelo de Domínio e o Metamodelo de IHC é discutido no Apêndice A.1. Os atributos foram mapeados para as respectivas questões: Nome (Questão Aberta), Data de Nascimento (Questão Aberta), CPF (Questão Aberta), Naturalidade (Agrupador) composto por Estado (Questão Fechada) e Cidade (Questão Fechada), e Idade (Questão Aberta).

Nota-se que, no elemento de negócio, uma Cidade deve pertencer a determinado Estado. É necessário colocar um Evento que monitore a seleção do Estado, para que então

seja apresentado ao usuário o conjunto de Cidades pertencentes a ele. Logo, é adicionada à interface a ação CarregaCidade, que será responsável por observar o Estado selecionado e disponibilizar as Cidades que pertencem ao Estado.

As demais ações associadas ao estereótipo Formulário já foram consideradas na Seção 3.2.3.

3.2.5 Correlação entre Apresentação e Comportamento de IU

Tanto a apresentação quanto o comportamento da interface são modelados de forma abstrata, através de metadados. Porém, estes aspectos não podem ser considerados isoladamente. Cada Elemento de IHC pode ter diferentes formas de interação. Entretanto, o tipo de elemento determina qual comportamento deve ser executado. Para cada Elemento de IHC haverá um conjunto de Eventos de Elemento que disparam as ações padrão do elemento.

Por exemplo, um elemento do tipo questão pode ter como padrão o evento "Mudança", e a este evento estar associada uma ação externa associada a uma regra de validação do Domínio de Negócio. Isto significa que em todas as instâncias do elemento de IHC questão que acontecer o evento "Mudança" será executada a ação de validação.

Porém, para um Elemento de IHC presente em um Estereótipo, ou seja, um Elemento Estereotipado, pode haver um conjunto de Eventos Estereotipados cuja ocorrência pode alterar a ação a ser disparada, possibilitando estender a definição do comportamento para o Elemento Estereotipado. Por exemplo, para o evento "Mudança" citado acima, pode ser alterada a ação que deve ser disparada para uma ação externa nomeada como "Calcular".

Assim, se um Elemento de IHC pertence a vários Estereótipos, tem-se vários Elementos Estereotipados, e estes, por sua vez, podem ter comportamentos variados, de acordo com a especificação do estereótipo. Por exemplo, um elemento do tipo questão aberta em um estereótipo formulário dispara uma ação de preencher um outro elemento do tipo questão. Já este mesmo elemento questão aberta, em um estereótipo de Login, dispara uma ação de validação.

Outra consideração importante é que os eventos de entrada de dados acontecem nos elementos questão aberta e questão fechada. Isto é, um elemento questão aberta ou fechada requer que o usuário realize uma interação que transfere informações para a interface do sistema. A Resposta dada pelo usuário nestes elementos ocorre por meio dos eventos de entrada de dados pelo usuário em resposta a uma questão aberta ou fechada. Já os eventos de solicitação ocorrem em elementos de informação do tipo serviço, uma vez que este tipo de elemento aguarda que o usuário acione-o para disparar a execução de uma ação.

Os demais tipos de elemento de Informação não têm por objetivo habilitar eventos para realização de ações externas, uma vez que apenas comunicam informações do sistema. Entretanto, estes tipos de elemento podem habilitar ações de transição de estados e de navegação, já que estas ações apenas auxiliarão o usuário a encontrar as informações desejadas. Por exemplo, um trecho de texto que contenha ligações que conduzem a outras partes do texto (hipertexto).

O conjunto de estados de uma IU deve ser especificado pelo seu Estereótipo, ou seja, todas as IU que são instâncias do mesmo Estereótipo devem possuir o mesmo conjunto de estados, e as mesmas ações. A diferença está na implementação das ações externas, com o objetivo de que cada instância de determinado Estereótipo implemente as ações especificadas com as particularidades requeridas pela respectiva aplicação.

Como exemplo, temos as ações do estereótipo Formulário, apresentado na Seção 3.2.3, que possui as ações internas Cancelar e Limpar com comportamento comum a todas as instâncias de Formulário, e a ação Salvar que terá uma implementação para as particularidades de cada instância de Formulário.

Os metadados do Metamodelo de IHC são abstratos o suficiente para que sejam produzidas descrições de interfaces independente de uma tecnologia. Para que seja gerada uma interface, existem informações específicas da plataforma computacional que devem ser consideradas. Estas questões são tratadas em outro metamodelo, apresentado a seguir.

3.3 Metamodelo de Apresentação

O objetivo do Metamodelo de Apresentação é representar a Interface Concreta, isto é, os elementos de interface específicos para uma determinada plataforma. Este metamodelo estende o Metamodelo de Interface proposto em [17], possibilitando a geração de outros tipos de interface, além de interfaces do tipo CRUD. Assim como o metamodelo original, a extensão também é específica para IU baseadas em WIMP.

A Figura 3.6 apresenta o Metamodelo de Apresentação para SI. Este metamodelo é composto por **Templates**, que são a representação concreta dos Estereótipos de Interface em determinada plataforma. O pacote Aplicação, externo ao metamodelo, representa a lógica de negócio das aplicações de SI. O mapeamento entre o Metamodelo de IHC e o Metamodelo de Apresentação é exibido no Apêndice A.2.

Um Template é conhecido como uma descrição visual da apresentação de uma aplicação ou documento, padronizando a aparência [76]. Neste trabalho este conceito vai além, representando não apenas a apresentação visual de interfaces com o mesmo propósito, como também o comportamento esperado desta interface.

Um Template possui um Idioma que o descreve. Uma mesma aplicação pode ser executada em diferentes **Idiomas**. Para que a interface de usuário possa ser apresentada

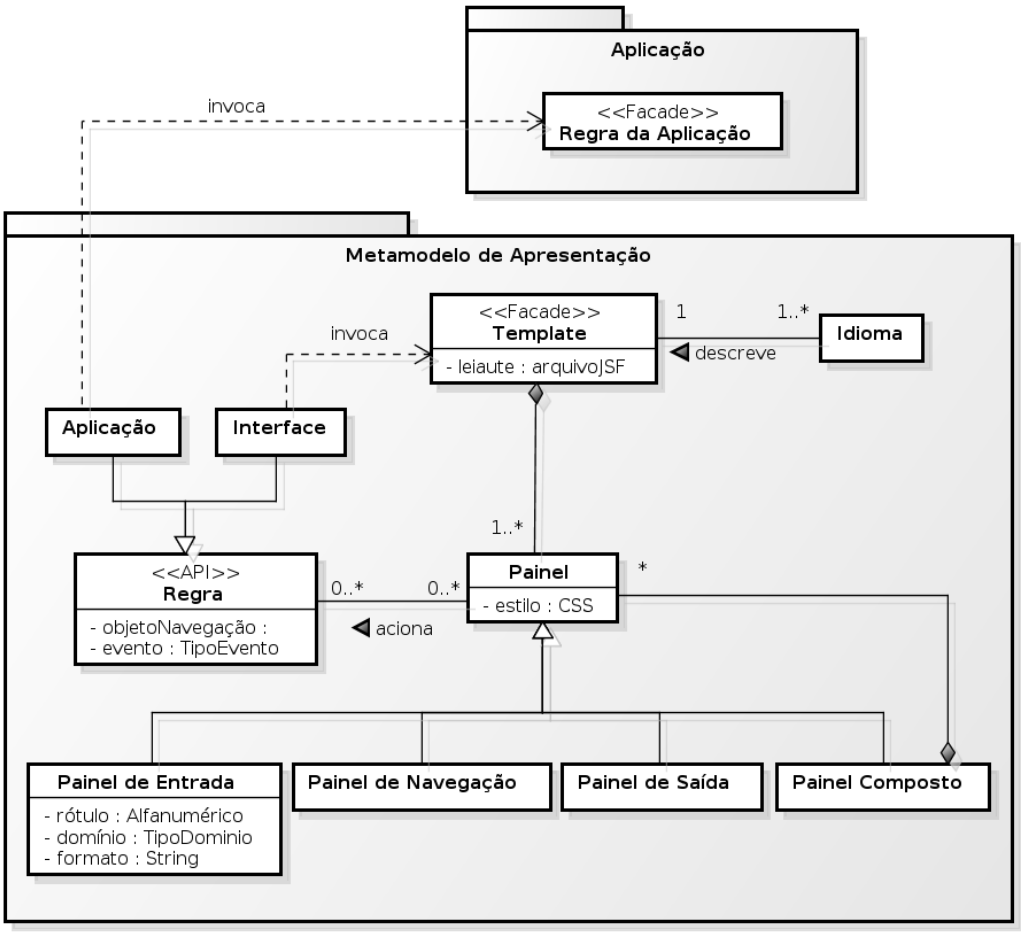


Figura 3.6: *Metamodelo de IU concreta*

em diversos idiomas, o Template deve prover apoio para internacionalização, de acordo com o idioma da aplicação subjacente.

Uma vez que os metadados do domínio de negócio são descritos em diversos idiomas, o template deve identificar em qual deles a aplicação deve ser executada. Assim, o template deve enviar esta informação para o domínio de negócio, a fim de que os dados do domínio sejam repassados no idioma adequado.

Este mecanismo de apoio à internacionalização das aplicações geradas é prático, por fornecer apoio à geração de aplicações multi-idioma de maneira simples.

Um Template é uma composição de Paineis. **Painel** é o conceito concreto para os Elementos de Interação da interface que devem transmitir informações ou receber interações do usuário. Existem quatro tipos de Painel: **Painel de Entrada**, **Painel de Navegação**, **Painel de Saída** e **Painel Composto**, de forma que é possível representar componentes de variados tipos.

Painel de entrada é o elemento que recebe informações do usuário para serem armazenadas na aplicação de SI. Ele ainda deve ser restringido pela cardinalidade, formato e domínio. Isto é, estas informações, contidas no Domínio do Negócio e mapeadas para

elemento de IHC, auxiliam na implementação de paineis que não permitem a entrada de dados que não pertencem ao domínio, formato e cardinalidade requeridos pelo contexto de negócio. Deste modo, é atendido o critério de usabilidade de proteção a erros [3], por ser um mecanismo que evita erros de entrada de dados por parte dos usuários.

O painel de navegação aciona uma ação após a interação do usuário. Painel de saída apresenta informações ao usuário e não esperam qualquer interação. O painel composto é uma composição de vários painéis.

Conceitos de Estilo e Leiaute devem ser considerados em nível concreto, e para isto eles foram associados ao conceito de Painel. Estilo é a descrição da aparência de determinado painel em termos de tamanho (em porcentagem), tipo de fonte, cor, cor de fundo, bordas, entre outros conceitos. O estilo deve ser descrito no formato CSS (*Cascating Style Sheets*) [75]. O leiaute está relacionado com o posicionamento dos paineis na interface. Um arquivo JSF deve ter a posição de cada painel, e informações em CSS complementam a descrição do leiaute.

Há dois tipos de **Regras** que devem surgir do mapeamento do conceito de Ação do nível abstrato: **Regras de Interface** e **Regras de Aplicação**. As regras de interface são aquelas que independem da lógica da aplicação para serem executadas, e as regras de aplicação são aquelas que serão implementadas pela aplicação de SI.

As regras são definidas através de uma API (*Application Programming Interface*). Elas devem encapsular um **Objeto de Navegação** que conduz as informações necessárias da interface para a regra de aplicação a ser executada. Este objeto deve conter o elemento de negócio manipulado na interface, bem como as informações requeridas pela aplicação para que as regras sejam executadas adequadamente. Por exemplo, para que uma regra de transição de interfaces seja executada adequadamente, é necessário enviar a informação de qual será a próxima interface a ser executada.

O comportamento é, então, especificado conceitualmente, através de regras que definem uma API, sem especificar a sua forma de implementação. As regras devem ser implementadas pela aplicação ou sistemas externos, que devem implementar a fachada definida pelo pacote Aplicação para as regras solicitadas pela interface. Isto garante a existência de um comportamento para cada interação que requer uma funcionalidade da aplicação, sem determinar como será a implementação deste comportamento pela aplicação.

A implementação das regras de aplicação deve ser realizada pela aplicação ou sistemas externos, de forma semelhante ao mecanismo de regras de negócio proposto por [2]. Dentre as regras de aplicação, destacam-se as Regras de Validação (conforme a taxonomia de [69] e a definição do framework de gestão de SI [2]).

Para a implementação das regras de interface, o Template oferece uma interface que manipula estas regras, implementando o padrão de projeto *Façade*, facilitando a

utilização das regras de interface.

A aplicação ou o template implementam as regras definidas na API do estereótipo, e estas regras devem ser invocadas por meio de uma fachada, fazendo com que a lógica de IU fique isolada da aplicação. Isto torna a geração de SI integrada ao ponto de vista da IHC e, ao mesmo tempo, desacoplada, uma vez que cada aspecto do sistema (interface e aplicação) é tratado de maneira separada.

Dessa forma, o comportamento da interface pode ser relacionado com a lógica de negócio da aplicação subjacente. Cada tipo de template deve implementar a classe Template, definindo seus elementos e associando-os ao comportamento descrito na API de regras por meio da fachada. Essa API pode ser implementada pela aplicação subjacente por meio da fachada Regras de Aplicação (que define a *Lógica de Negócio*).

A descrição do comportamento por meio de uma API torna a integração da interface de usuário com a lógica de negócio uma tarefa mais simples de ser realizada. Isto ocorre por meio da implementação do padrão de projeto Façade, que define uma interface de nível mais alto para as regras.

A Figura 3.7 exibe um exemplo de aplicação do Metamodelo de Apresentação, que é a representação concreta dos dados abstratos da Figura 3.5. Em relação ao Metamodelo de IHC, este exemplo apresenta a classe CadastraPessoa, que é uma implementação de Template. Esta classe possui a implementação dos métodos necessários para executar as funcionalidades requeridas pelo usuário: limpar e cancelar, que são regras de interface, e salvar, que é uma regra de aplicação invocada pelo template. Ainda para o caso específico do exemplo em questão, há a regra de aplicação Carrega Cidade, que é invocada por meio do painel Estado.

Com a descrição concreta da interface, deve existir um mecanismo que interprete-a a fim de gerar a interface final para o usuário. Isto requer uma arquitetura de software que apoie o emprego dos metamodelos aqui propostos. O próximo capítulo apresenta a arquitetura de um componente para gerenciar o processo de construção de IU baseado nos metamodelos descritos neste capítulo.

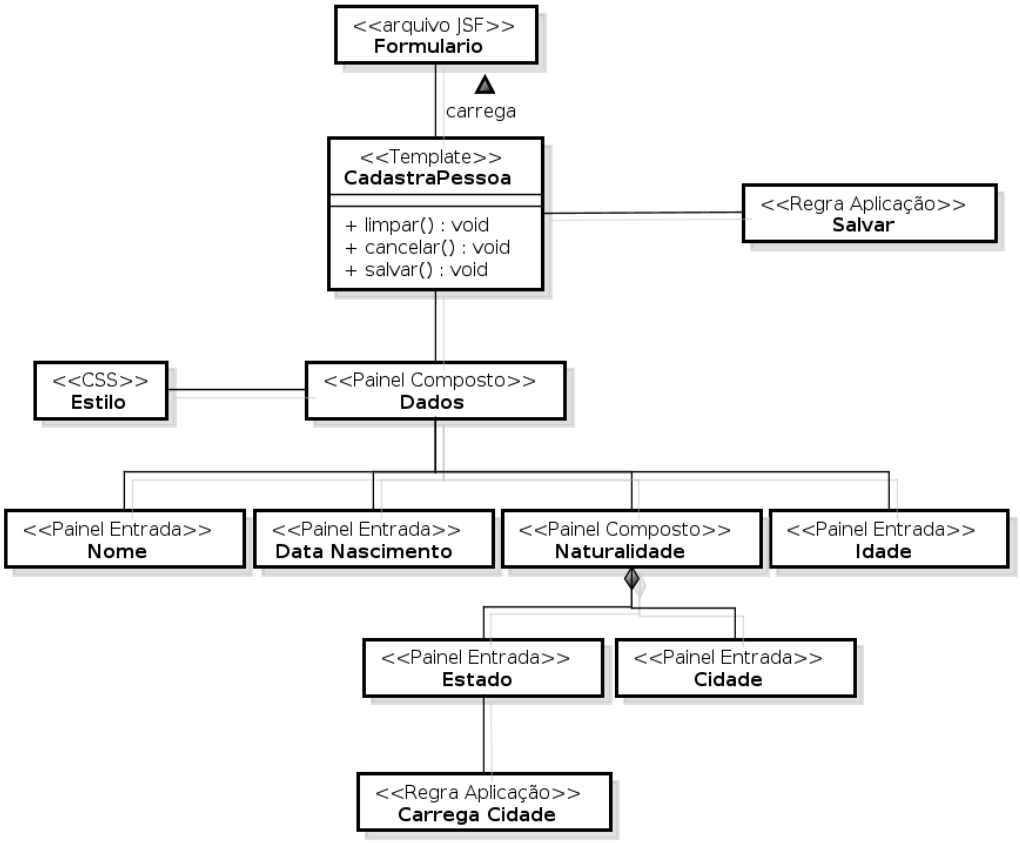


Figura 3.7: *Exemplo de Aplicação do Metamodelo de Apresentação*

CAPÍTULO 4

Arquitetura para um Sistema de Gerência de IU

A arquitetura para gerência de IU proposta neste capítulo traz um conjunto de visões arquiteturais com o objetivo de descrever os diferentes aspectos de um componente de gerência de IU. Esta arquitetura supera algumas limitações das abordagens atuais, de modo a administrar a complexidade da construção de IU baseada em modelos para SI.

Esta solução arquitetural faz parte de uma macro-arquitetura de *framework* baseado em modelos para gestão de SI, onde diferentes aspectos de SI são considerados [2]. Uma visão geral desta macro-arquitetura é apresentada na Seção 4.1. A interface de usuário é um destes aspectos. Porém estes aspectos devem ser integrados para que formem um Sistema de Informação executável.

Uma visão geral da arquitetura para gerência de IU é apresentada na Seção 4.2. A Seção 4.3 explana a Visão Lógica do módulo Modelo, mostrando como os metamodelos apresentados no Capítulo 3 são empregados para a geração automática de IU. Finalmente a Seção 4.4 apresenta a forma de comunicação entre os pacotes do módulo Modelo.

4.1 Visão geral da Arquitetura do *Framework* de Gestão de SI

O Grupo de Pesquisa em Engenharia de Software do Instituto de Informática (INF) da Universidade Federal de Goiás tem investido um grande esforço de pesquisa no sentido de construir componentes para apoiar o desenvolvimento de software que contemplem os diversos aspectos de Sistemas de Informação. Neste sentido, um *framework* para desenvolvimento de software específico de domínio foi construído para apoiar a síntese de SI baseado em modelos de alto-nível [2, 15, 34].

O *framework* para gestão de SI desenvolvido pelo Grupo de Pesquisa do INF teve uma primeira versão [2] e atualmente encontra-se em evolução, já que algumas dificuldades foram encontradas [25].

A ideia básica deste *framework* é que existem três aspectos primordiais para o projeto de SI, a saber: dados, processos e interface de usuário [15], e estes podem ser

tratados de maneira separada, por especialistas que tratam das peculiaridades de cada aspecto. Porém, eles devem ser integrados de forma adequada formando um Sistema de Informação. A Figura 4.1 apresenta os componentes deste *framework*.

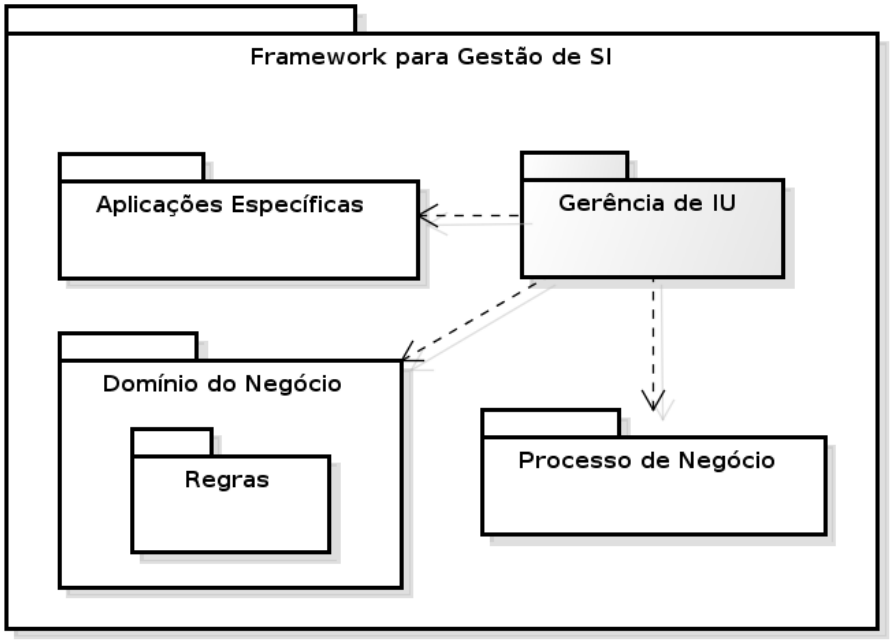


Figura 4.1: *Visão Geral da Arquitetura do Framework para Gestão de SI*

Na Figura 4.1 observa-se que cada um dos componentes do *framework* trata um aspecto de um Sistema de Informação. Como pode-se observar, um Sistema de Informação é composto por um Domínio de Negócio, que compreende os dados do negócio e as regras associadas que serão modificadas e aplicadas pelo SI. As regras são tratadas pelo mecanismo proposto em [2].

Um Sistema de Informação também contém um Processo de Negócio, que organiza as ações e tarefas para gerar e modificar dados de negócio, melhorando a compreensão das atividades e seus relacionamentos. Estas são aplicações predefinidas para SI, ou seja, qualquer Sistema de Informação compreende estas duas aplicações. Além disso, SI contêm Aplicações Específicas do negócio, já que dependendo do domínio da aplicação podem surgir necessidades específicas que devem ser consideradas.

Deste modo, este trabalho propõe o componente para Gerência de IU, que utiliza as informações contidas nos demais aspectos de SI para gerar interfaces e acionar funcionalidades. O restante deste capítulo descreve as características deste componente.

4.2 Visão geral da Arquitetura do componente para Gerência de IU

Para que a abordagem automatizada para construção de IU seja eficiente, é necessário uma ferramenta que possibilite a geração de IU a partir dos Metamodelos discutidos no Capítulo 3. A utilização de metamodelos em diferentes níveis de abstração possibilita a extração de informações suficientes para que seja possível a geração automatizada de IU, da mesma forma que ocorre na abordagem [46], em que são definidos metamodelos em dois níveis de abstração.

O primeiro nível contempla metadados abstratos, que representam particularidades do domínio, e o segundo nível abrange informações da plataforma computacional, que serve de base para a aplicação.

A arquitetura proposta para o gerenciador de IU utiliza estes dois níveis de abstração: Interface Abstrata, que compreende os metadados do Metamodelo de IHC; e Interface Concreta, que contempla uma descrição com detalhes computacionais.

O componente de Gerência de IU emprega o padrão arquitetural MVC (*Model-View-Controller*) [29], conforme mostra a Figura 4.2.

Os pacotes Metamodelo de IHC e Metamodelo de Apresentação têm a responsabilidade de manipular os Metamodelos de IHC e de Apresentação respectivamente. Já o pacote Transformador Domínio-Interface Abstrata utiliza uma instância do Metamodelo de Domínio de Negócio como entrada, aplica um conjunto de regras de transformação e gera uma instância do Metamodelo de IHC, que corresponde à uma descrição abstrata da interface de usuário. Analogamente, no pacote Transformador Interface Abstrata-Interface Concreta transforma-se uma instância do Metamodelo de IHC em uma instância do Metamodelo de Apresentação.

Para que a interface real seja executada pela aplicação, foi criado o pacote Interpretador Interface Final, que utiliza uma instância do Metamodelo de Apresentação para gerar os componentes gráficos em uma linguagem computacional. Estes componentes gráficos devem ser fornecidos por uma Biblioteca de Componentes. Exemplos deste tipo de biblioteca são os pacotes Swing [49] e AWT [48] (em Java, para aplicações *desktop*), OpenFaces [67] e RichFaces [11] (em JSF, para aplicações *Web*).

O pacote Controlador trata as requisições do cliente, e também seleciona a visão correta para apresentar ao usuário. Para cada solicitação do usuário de manipulação, ou mesmo de transformação de modelos, é o controlador o responsável por interpretar a solicitação, efetuar a chamada de métodos do modelo e enviar corretamente os dados para a visão.

No pacote controlador, a classe DominioAbstrataControle é responsável por tratar as requisições para transformar os dados do Domínio do Negócio em Interface

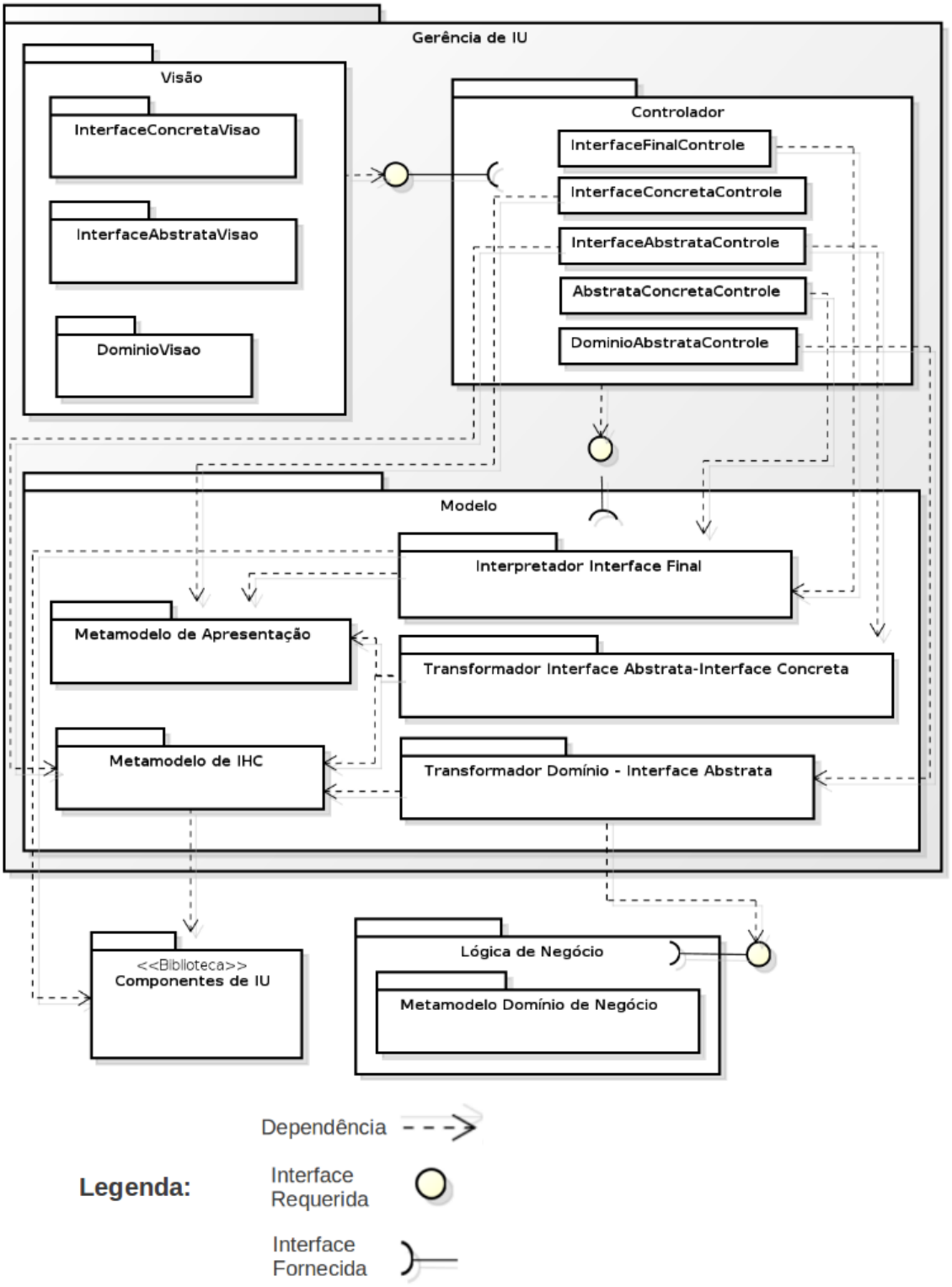


Figura 4.2: *Visão Geral da Arquitetura para gerência de IU*

Abstrata através do Transformador Domínio - Interface Abstrata. Analogamente, a classe AbstrataConcretaControle trata as requisições para transformar a Interface Abstrata para Concreta pelo Transformador Interface Abstrata - Concreta.

As classes InterfaceAbstrataControle e InterfaceConcretaControle tratam as re-

quisições para manipulação dos dados do Metamodelo de IHC e de Apresentação respectivamente. Já a classe InterfaceFinal trata as requisições em tempo de execução para executar as interfaces por meio do pacote Interpretador Interface Final.

Por fim, o pacote Visão possui as IU do componente, onde os metadados podem ser manipulados. Uma vez selecionada, uma interface lê os dados do objeto do metamodelo manipulados pelo controlador e gera uma resposta. A resposta é exibida na interface de usuário. O pacote InterfaceConcretaVisao possui as interfaces que manipulam os dados do Metamodelo de Apresentação que foram gerados e permite altera-los conforme a necessidade do projetista.

O pacote InterfaceAbstrataVisao possui as interfaces para manipulação do Metamodelo de IHC que foram gerados pelo Transformador Domínio - Interface Abstrata ou que foram inseridas pelo projetista. Da mesma forma, o pacote DominioVisao possui as interfaces que apresentam os dados do domínio do negócio e possibilitam requisitar a transformação para Interface Abstrata.

O componente para gerência de IU faz parte do *framework* para gestão de SI. Este *framework* é implantado em um ambiente onde o usuário interage com a interface para construir SI. Quando o usuário solicitar alguma funcionalidade do componente de gerência de IU, este deve interpretar a solicitação e, caso necessário, realizar a comunicação com o SGBD para persistir ou consultar algum dado. O componente de gerência de IU depende das informações da lógica de negócio para trabalhar adequadamente no processo de construção de IU.

4.3 Visão Lógica do Pacote Modelo

O pacote Modelo foi produzido com o objetivo de manipular os metadados de IU e transformar estes metadados em interfaces executáveis. Seguindo os passos de desenvolvimento do framework CAMELEON [9], foram definidos os Metamodelos de IHC e de Apresentação no Capítulo 3 que descrevem a interface em diferentes níveis de abstração e capturam os metadados necessários para a geração de IU automatizada.

4.3.1 Pacote Metamodelo de IHC

A Figura 4.3 exibe um Diagrama de Classes que ilustra o pacote Metamodelo de IHC. Este pacote tem o objetivo de manipular os dados do Metamodelo de IHC, que foi definido conceitualmente na Seção 3.2.

A Classe **Estereótipo** contém o método **obterMetadados** que permite recuperar todos os metadados de uma interface abstrata. Por meio do parâmetro *idEst* (id do estereótipo dominante da interface) são obtidos os metadados do estereótipo dominante,

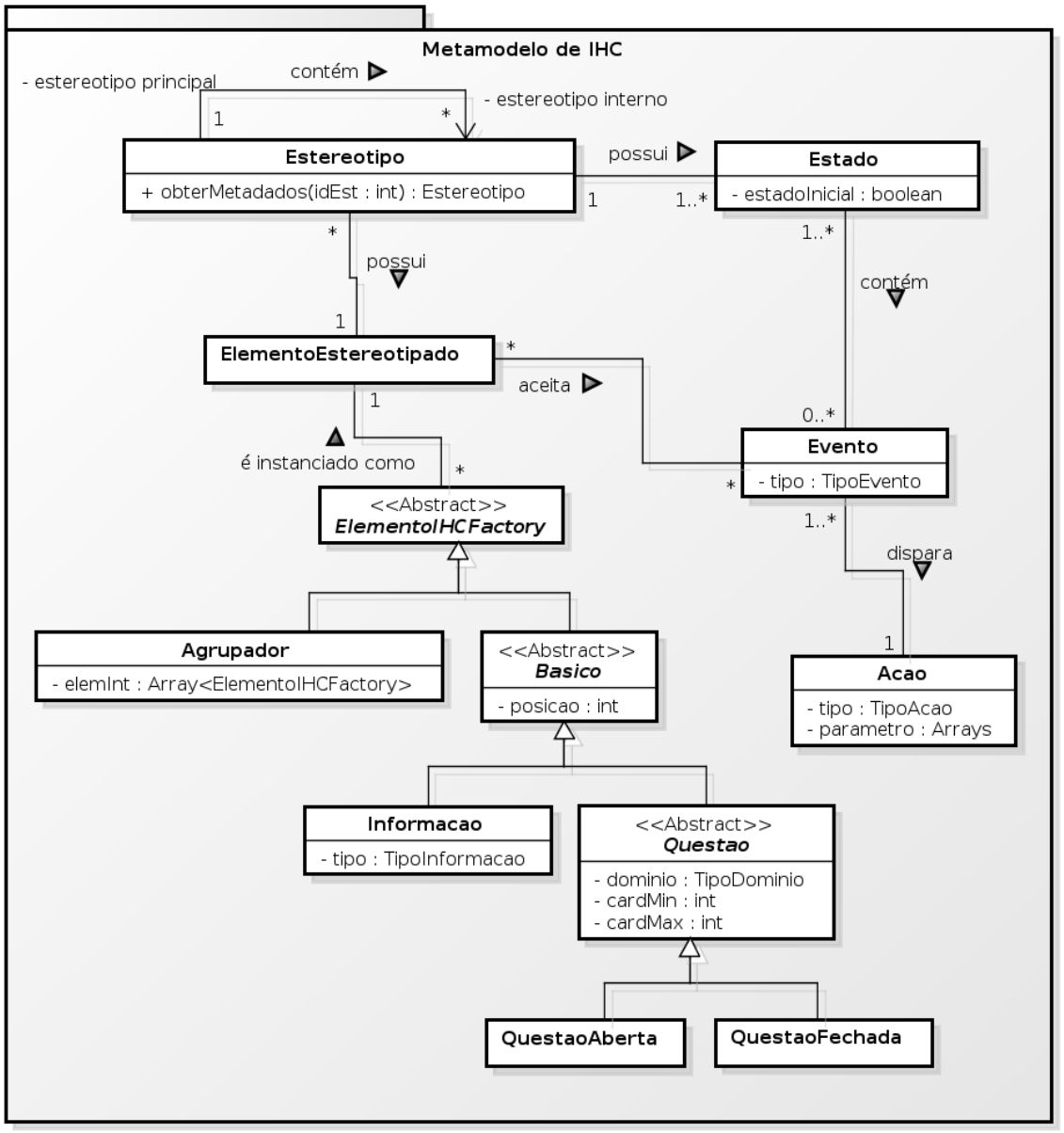


Figura 4.3: *Visão Lógica do Metamodelo de IHC*

e dos estereótipos que o compõe, juntamente com seus elementos de interação e ações da interface registrada. Este método deve retornar um objeto Estereotipo com todos os seus metadados.

A classe ElementoIHCFactory implementa o padrão de projeto *Factory* [24]. As classes **Agrupador** e **Basico** são especializações de ElementoIHCFactory. Um Agrupador é uma composição de ElementoIHCFactory, que é representado pela propriedade elemInt, que é um *Array* de ElementoIHC, enquanto que Basico tem a propriedade posicao, representando a posição do componente no estereótipo ou no agrupador em que ele se encontra.

A classe **Questao** possui as propriedades domínio, cardinalidade mínima e cardinalidade máxima. A classe **Informacao** possui a propriedade tipo, que identifica o

tipo de informação que deve ser apresentada, conforme os tipos definidos na Seção 3.2.1.

Questao ainda se especializa em **QuestaoFechada** e **QuestaoAberta**. QuestaoFechada deve ter como valor o tipo Enumerado para a propriedade de domínio herdada da superclasse Questao.

A classe **Estado** tem a propriedade estadoInicial, que deve indicar um único estado de um estereótipo como estado inicial da interface. A classe **Evento** deve ter a propriedade tipo de evento, conforme a taxonomia definida na Seção 3.2.2. A classe Ação deve ter as propriedades de tipo, conforme definido na Seção 3.2.2 e parametro, que consiste em uma lista que representa os parâmetros que uma ação pode ter. Esta lista de parâmetros é utilizada para auxiliar na implementação das ações.

4.3.2 Pacote Metamodelo de Apresentação

O pacote Metamodelo de Apresentação é responsável por manipular os dados do Metamodelo de Apresentação, discutido na Seção 3.3. Ele é representado pelo Diagrama de Classes apresentado na Figura 4.4. A classe **Template** representa os metadados concretos de uma interface, que pode ser uma composição de templates.

A classe **Template** é uma classe abstrata, possibilitando que cada tipo de template herde suas funcionalidades e propriedades. Ela tem como propriedade objDados, que é do tipo Elemento de Negócio, isto é, ela encapsula um conjunto de Metadados de Contexto de Negócio do Metamodelo de Domínio para manipulação na interface.

A classe **Painel** contém as propriedades estilo, que é o nome da classe em CSS (Cascading Style Sheets) que contém os detalhes gráficos para o painel, e posição, que é a informação de posicionamento do painel na interface.

As classes **Navegacao**, **Saida**, **Entrada** e **Composto** são baseadas nos componentes gráficos para interfaces WIMP da Biblioteca OpenFaces [67], de forma que estas classes manipulam os metadados para gerar os componentes nesta biblioteca.

Para cada tipo de Template, deve existir uma *Interface* com as regras que a aplicação deve implementar. *Interfaces* servem como um contrato, fazendo com que ela seja o meio de relacionar a lógica de negócio à descrição da interface. Além disso, o próprio template implementa as regras de interface com a implementação do padrão *Façade*.

4.3.3 Pacote Transformador Domínio para Interface Abstrata

A arquitetura deste pacote foi projetada para realizar a transformação entre o Metamodelo de Domínio de Negócio e o Metamodelo de IHC. Isto implica em realizar o mapeamento da descrição dos dados do domínio para uma forma de apresentação de acordo com um estereótipo.

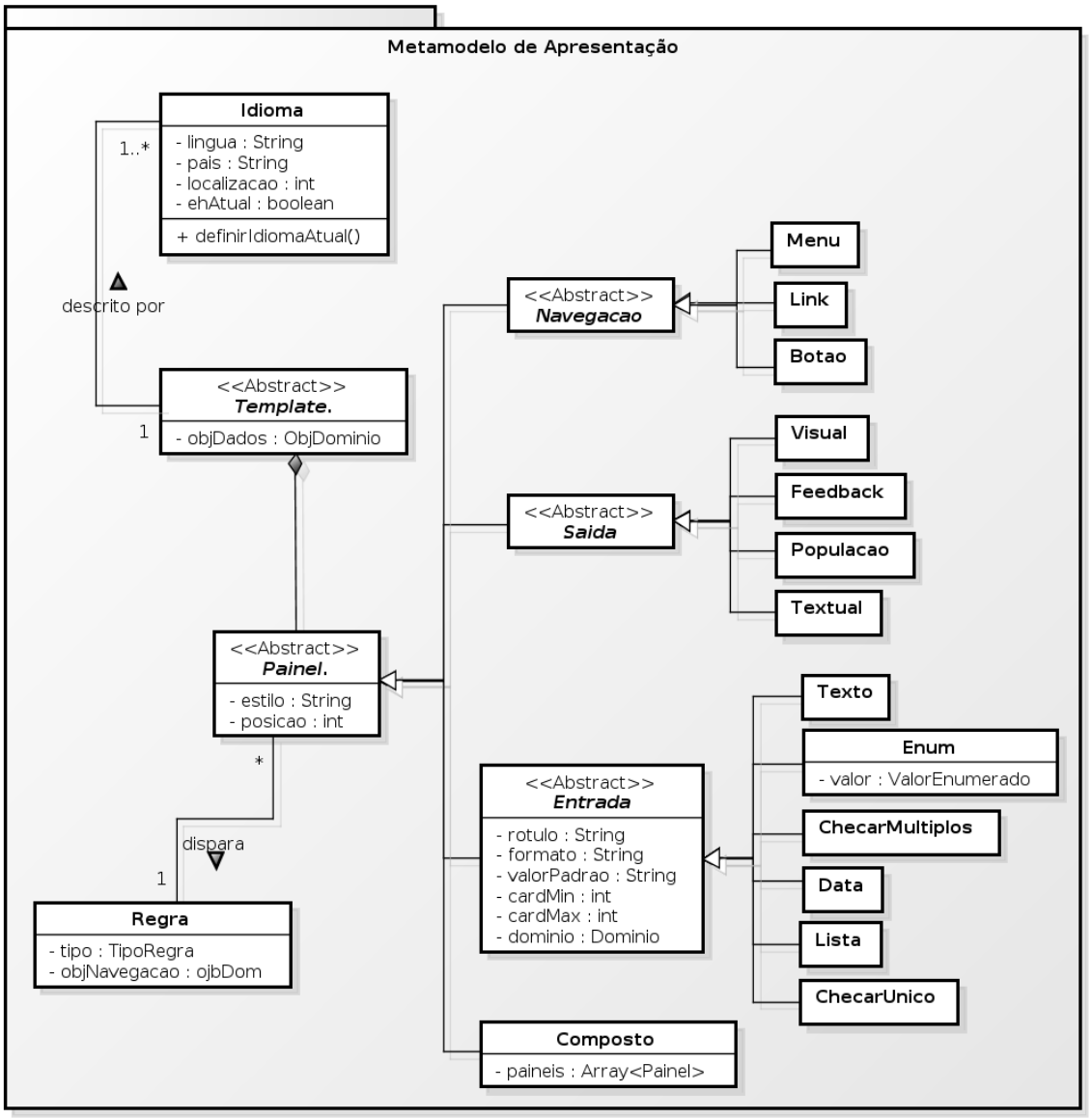


Figura 4.4: *Visão Lógica do Metamodelo de Apresentação*

Com base no Metamodelo de Domínio, o transformador recebe um ObjetoDominio contendo os metadados do contexto de negócio que serão manipulados pela interface. A partir dele, são realizados mapeamentos gerando o Metamodelo de IHC.

Para este mapeamento, e com base em um conjunto de estereótipos de interface previamente modelados e implementados, é necessário informar o Estereótipo de Interface selecionado. Com esta informação é possível saber as funcionalidades que precisam ser implementadas e ainda os tipos de componentes que serão apresentados com as informações do domínio. Para isto, foram considerados os principais conceitos dos dois pacotes em questão:

- **Metamodelo de Domínio:** Elemento de Negócio e Atributo.
- **Metamodelo de IHC:** EstereotipoInterface e ElementoIHC.

Dessa forma, o tipo de Estereótipo de Interface é fundamental para a definição das transformações, e cada tipo de Estereótipo pode ter uma configuração diferenciada das regras de transformações. As regras de transformação para este pacote são resumidas na Tabela 4.1.

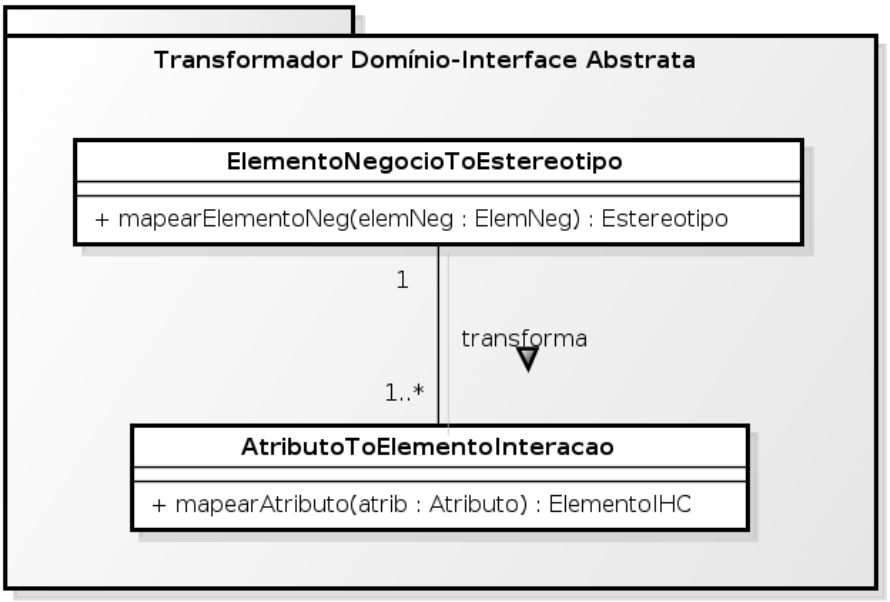


Figura 4.5: *Visão Lógica do Transformador Domínio para Interface Abstrata*

A Figura 4.5 exibe a Visão Lógica do pacote Transformador Domínio-Interface Abstrata. Nele existem duas classes: **ElementoNegocioToEstereotipo** e **AtributoToElementoInteracao**. A classe ElementoNegocioToEstereotipo contém o método **mapearElementoNeg**, que recebe como parâmetro um elemento de negócio e retorna uma instância de estereótipo.

A classe AtributoToElementoInteracao possui o método mapearAtributo, que recebe um atributo como parâmetro e retorna um elemento de IHC.

Tabela 4.1: *Regras de Mapeamento do pacote Transformador Domínio para Interface Abstrata*

Metamodelo de Domínio	***Metamodelo de IHC***
Elemento de Negócio	Estereótipo de Interface
Atributo de Elemento de Negócio	Elemento de IHC
Regra	Ação

Cada elemento de negócio pode ser mapeado para diferentes estereótipos, dependendo do propósito da interface. O estereótipo deve ser selecionado pelo projetista da aplicação. A partir dessa seleção, o restante das transformações é efetuada automaticamente por meio deste pacote.

Cada atributo do elemento de negócio deve ser mapeado para um elemento de IHC. De acordo com o estereótipo selecionado, haverá um conjunto de elementos de IHC disponíveis para o mapeamento. Caso ocorram problemas para realizar este mapeamento, um tratamento de exceção notifica o projetista sobre o problema, permitindo então que seja indicado o mapeamento manualmente. Por exemplo, para o estereótipo Formulário, cada atributo deve ser mapeado para um elemento de IHC do tipo Questão, já que o propósito deste estereótipo é a manipulação de dados.

Cada regra relacionada ao metadado de contexto de negócio é mapeada para uma ação externa predefinida pelo estereótipo. As regras de transformação específicas para o estereótipo Formulário são descritas no Apêndice A.1.

4.3.4 Pacote Transformador Interface Abstrata para Concreta

O Metamodelo de IHC, que representa uma interface abstrata, não possui informações suficientes para a geração de uma interface de usuário em linguagem computacional. Para isto, detalhes da plataforma computacional precisam ser adicionados.

Neste trabalho, o Metamodelo de Apresentação permite representar estes detalhes para IU WIMP *Web*, com o uso da especificação JSF [47].

Analogamente ao pacote Transformador Domínio para Interface Abstrata, o pacote Transformador Interface Abstrata para Concreta realiza o mapeamento do Metamodelo de IHC para o Metamodelo de Apresentação. Este pacote utiliza uma instância do Metamodelo de IHC como entrada. A partir de um conjunto de regras de transformação é gerada uma instância do Metamodelo de Apresentação. Para tratar este mapeamento, foram considerados os conceitos principais dos pacotes:

- **Metamodelo de IHC:** EstereotipoInterface e ElementoIHC.
- **Metamodelo de Apresentação:** Template e Painel.

A partir destes conceitos, foi definida a visão lógica do pacote Transformador Interface Abstrata para Concreta em um Diagrama de Classes que pode ser observado na Figura 4.6.

A classe **EstereotipoToTemplate** realiza as transformações entre um Estereótipo de Interface e um Template. O método **mapearEstereotipo** é responsável por executar esta tarefa, recebendo um Estereótipo de Interface como parâmetro e retornando um Template. O método adicionarInfoMapeamento acrescenta informações que não estão descritas no Metamodelo de IHC, como descrição de estilo e leiaute. O estilo é adicionado por meio de um arquivo com a descrição em formato .css (Cascading Style Sheets), que é um padrão de formatação para páginas *Web* [75]. Já o leiaute deve estar em um arquivo template em JSF, e ele deve fazer uso do estilo em CSS.

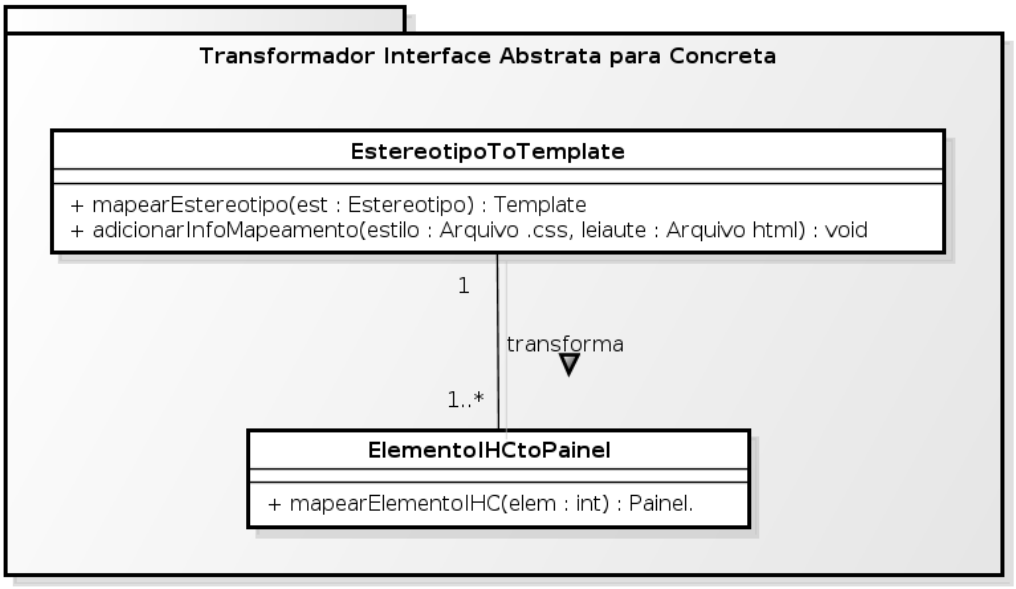


Figura 4.6: *Visão Lógica do Transformador Interface Abstrata para Concreta*

A classe **ElementoIHCToPainel** realiza o mapeamento entre um Elemento de Interação e um Painel. O método **mapearElementoIHC** realiza esta ação, recebendo um objeto do tipo ElementoInteracao e retornando um Painel. A Tabela 4.2 resume as principais regras de transformação necessárias para este pacote.

Tabela 4.2: *Regras de Mapeamento do pacote Transformador Interface Abstrata para Concreta*

Metamodelo de IHC	***Metamodelo de Apresentação***
Estereótipo de Interface	Template
Elemento de IHC	Painel
Ação	Regra

Mais detalhes das regras de transformação do Metamodelo de IHC para o Metamodelo de Apresentação são apresentadas no Apêndice A.2.

4.3.5 Pacote Interpretador de Interface Final

O pacote Interpretador de Interface de Usuário Final é responsável por ler os metadados concretos, através da obtenção de uma instância de Template, e interpretá-los em tempo de execução a fim de gerar IU em linguagem computacional. A implementação deste componente é fortemente dependente da biblioteca de componentes de IU selecionada para a geração das IU.

Cada Template deve ter uma representação da aparência na linguagem computacional desejada e do comportamento através da implementação de uma classe. A forma de representação depende da plataforma e do pacote de componentes de IU empregados.

Para plataforma *Web* utilizando a linguagem Java, com a especificação JSF e a biblioteca de componentes OpenFaces, a representação da aparência será em um arquivo JSF com extensão .xhtml empregando componentes do OpenFaces. A descrição do estilo e leiaute devem estar em arquivo CSS. Já o comportamento é implementado em uma classe do tipo *ManagedBean*, que estende a classe Template do pacote Metamodelo de Apresentação.

Assim, este pacote subdivide-se em dois subpacotes: *ManagedBeans* e *Interface Final*. O pacote *ManagedBeans*, apresentado na Figura 4.7, deve conter as classes que manipulam os dados do Metamodelo de Apresentação para geração das interfaces finais. Cada classe Template e os vários tipos de Paineis, são representados por um *ManagedBean* do JSF [47], que é responsável por intermediar a comunicação entre os dados do modelo e o código final da interface.

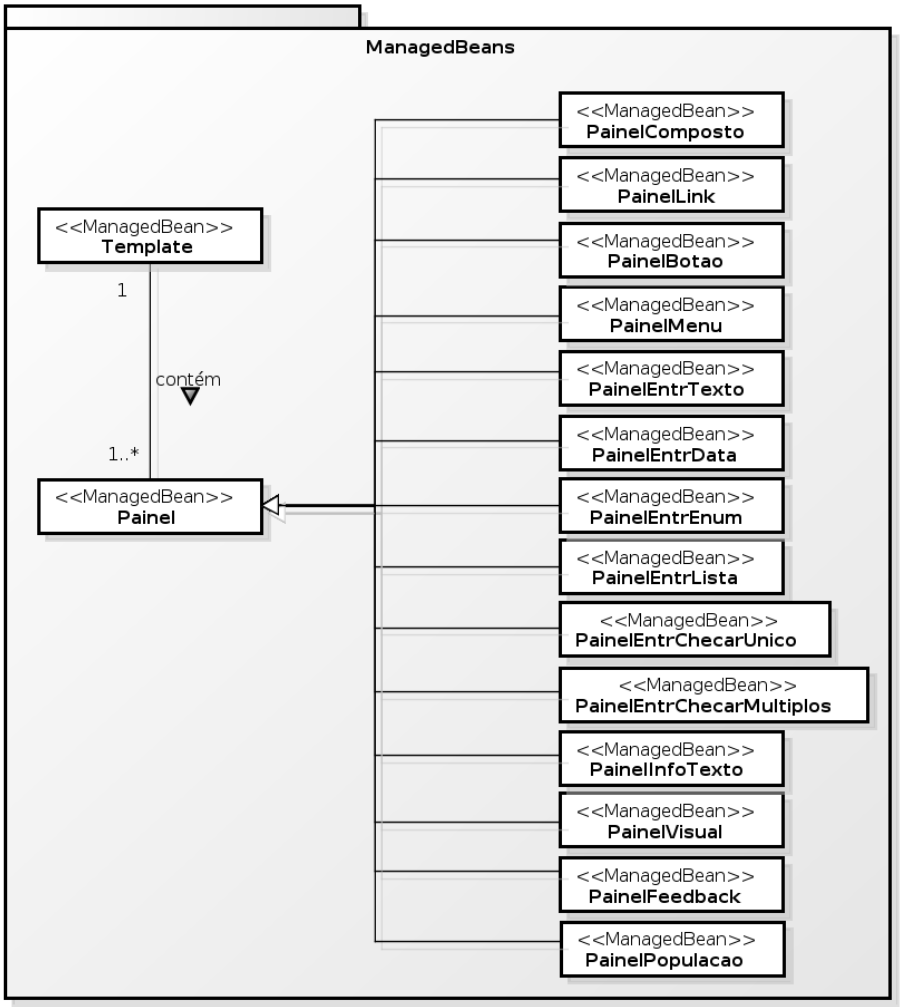


Figura 4.7: *Visão Lógica do Pacote ManagedBeans do Interpretador*

O pacote Interface Final, apresentado na Figura 4.8, exibe os componentes equivalentes a cada um dos *ManagedBeans* em forma de templates (para a classe Template) e

componentes JSF (para as especializações de Painel). Em cada arquivo de template JSF, o projetista deve implementar uma parte de painéis comuns a todas as instâncias do template, e uma parte que deve ser preenchida com os dados do elemento de negócio que foram mapeados pelos dois transformadores citados anteriormente. Se o template não estiver desta forma, uma exceção é gerada e tratada pelo Controlador.

Na modelagem do template, o projetista deve fazer um algoritmo para cada parte do template que deve conter sempre os mesmos tipos de elementos, mas com diferentes informações de domínio do negócio. Assim, cada template instanciado basicamente será preenchido com componentes JSF e OpenFaces que foram predeterminados pelo projetista, e outros componentes que são colocados dinamicamente de acordo com as informações vindas do domínio.

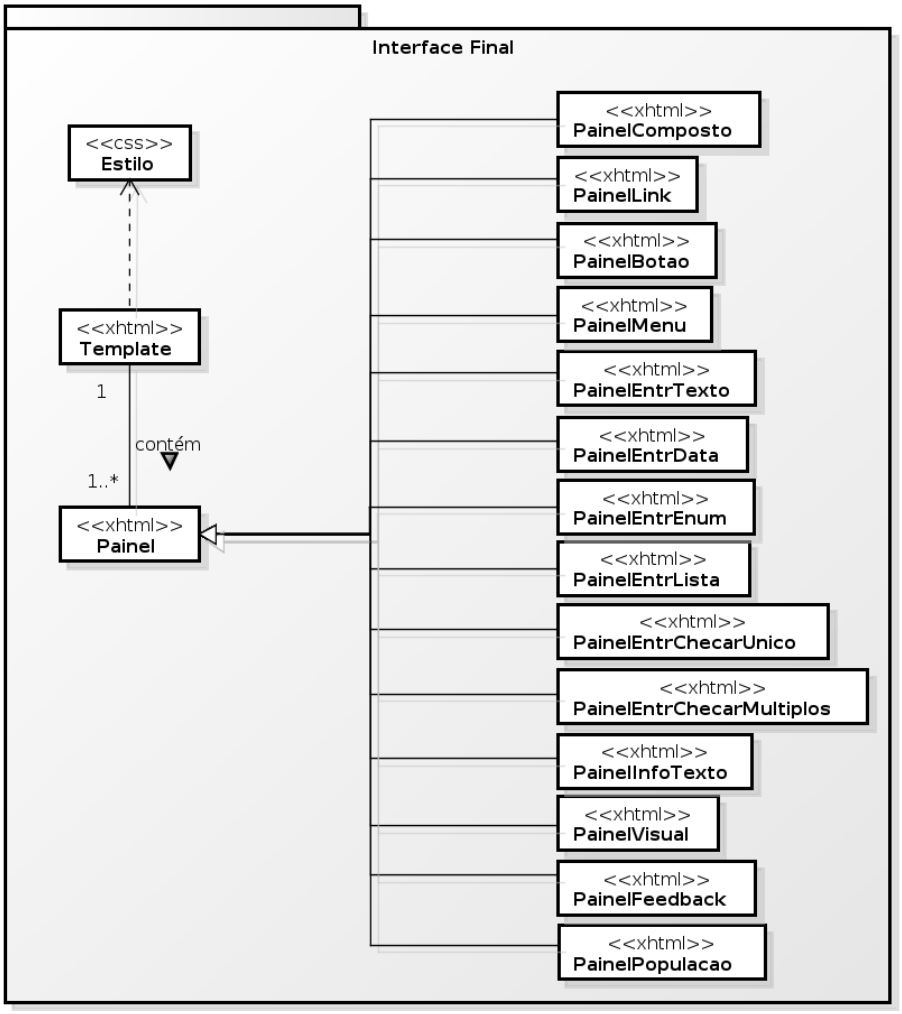


Figura 4.8: *Visão Lógica do Pacote Interface Final do Interpretador*

4.4 Comunicação entre os pacotes

Durante a fase de desenvolvimento das IU, os pacotes do Modelo do componente de gerência de IU se comunicam da forma observada na Figura 4.9. O projetista da aplicação que utiliza o componente solicita, através da interface de usuário do componente, a modelagem de uma interface informando o elemento de negócio e o estereótipo desejado, conforme mensagem 1 da Figura 4.9.

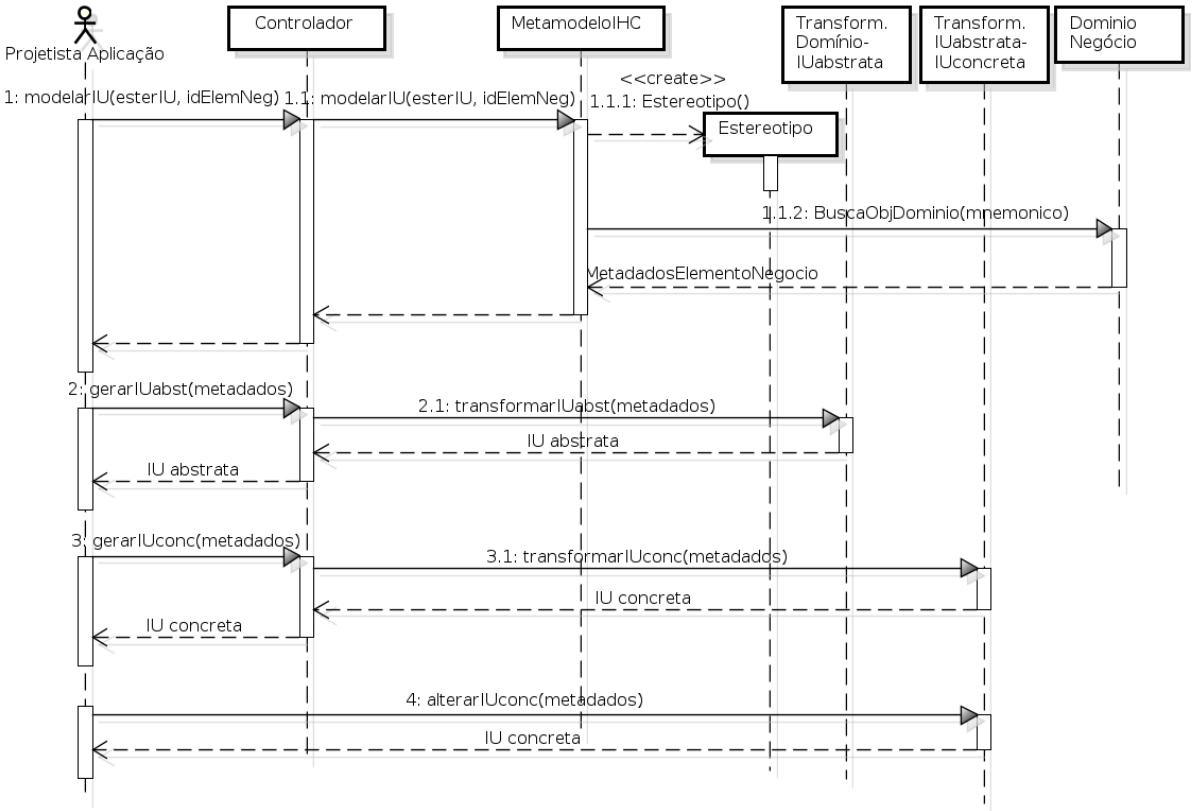


Figura 4.9: *Processo de Construção de IU utilizando o componente projetado*

O pacote Controlador envia uma requisição ao pacote Metamodelo IHC com as informações fornecidas (mensagem 1.1), onde é instanciado o tipo de estereótipo solicitado (mensagem 1.1.1). Este pacote também solicita ao pacote de Metamodelo de Domínio um objeto com os metadados do elemento de negócio requerido pelo projetista da aplicação (mensagem 1.1.2).

Com os metadados de elemento de negócio, o projetista da aplicação pode solicitar a geração da interface abstrata, de acordo com a mensagem 2. O pacote Controlador recebe esta requisição e envia para o pacote Transformador Domínio - Interface Abstrata, que retorna uma instância do Metamodelo de IHC.

O projetista deve então selecionar uma instância do Metamodelo de IHC para que seja mapeada para o Metamodelo de Apresentação. O pacote Controlador recebe

esta requisição (mensagem 3) e envia-a ao pacote Transformador Interface Abstrata para Concreta. Este pacote retorna uma instância do Metamodelo de Apresentação. Este modelo pode ainda ser manipulado pelo projetista, possibilitando pequenas alterações conforme as preferências dos usuários, e ainda adicionar informações detalhadas de estilo e leiaute que não são obtidas pelo metamodelo abstrato (mensagem 4).

Tendo o metamodelo de apresentação definido, a interface está pronta para ser executada pelo pacote Interpretador de Interface Final.

A Figura 4.10 exibe a forma de comunicação entre os pacotes durante a execução do Sistema de Informação. Para cada seleção do usuário para alterar a interface, indicando a solicitação de uma nova tarefa da aplicação, o pacote Controlador deve conduzir a uma interface diferente. Estas solicitações são enviadas ao pacote Controlador (mensagem 1 da Figura 4.10), que solicita os metadados concretos da interface desejada ao pacote Metamodelo de Apresentação (mensagem 1.1).

Os metadados concretos da interface desejada são enviados ao pacote Interpretador de Interface Final (mensagem 1.2). Neste pacote, os metadados concretos são lidos e gera-se os componentes finais executáveis em JSF. Estes componentes são carregados na interface de usuário do SI.

O controle da geração torna-se mais simples pela utilização da especificação JSF. Tendo as classes do pacote Metamodelo de Apresentação como beans gerenciáveis, a geração pode ocorrer de maneira dinâmica, bastando atualizar os metadados dos *beans* para a geração de uma nova interface.

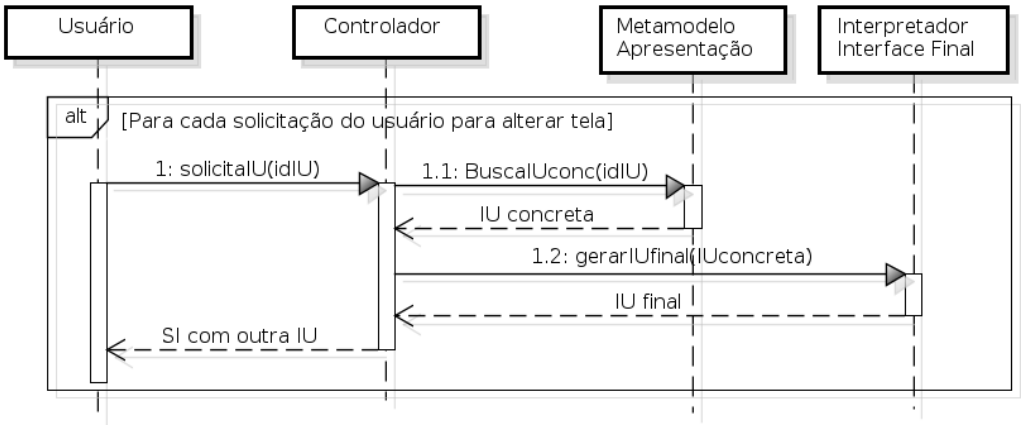


Figura 4.10: *Processo de Execução de IU utilizando o componente projetado*

A arquitetura para o componente de gerência de IU propõe assistir a abordagem de construção de IU proposta por este trabalho. É necessário avaliar se a abordagem atinge seu objetivo e qual a forma de aplicá-la. Deste modo, o próximo capítulo apresenta uma avaliação desta abordagem, mostrando exemplos de estereótipos e como utilizá-los nesta abordagem.

CAPÍTULO 5

Construção Automática de IU Baseada em Modelos

Este capítulo discute a utilização da abordagem para construção de IU baseada em modelos proposta neste trabalho. A Seção 5.1 apresenta os cenários que ilustram a utilização da abordagem proposta. A Seção 5.2 analisa um cenário de uso específico de modelagem de IHC baseada em estereótipos identificados no contexto de SI. As seções 5.3, 5.4 e 5.5 discutem a transformação de modelos de IHC em interfaces reais utilizando os projetos deste trabalho. Finalmente a Seção 5.6 discute como integrar a abordagem proposta com o desenvolvimento de SI.

5.1 Visão Geral da Abordagem Proposta

A construção de IU com a abordagem aqui proposta envolve um conjunto de atividades para construção e evolução de IU com base no componente para Gerência de IU introduzido no Capítulo 4. A Figura 5.1 resume o processo de desenvolvimento proposto por esta abordagem em um Diagrama de Atividades.

O processo de desenvolvimento desta abordagem segue os principais passos do *framework* CAMELEON [9]: Modelagem de Estereótipos, Modelagem da Interface Abstrata, Modelagem da Interface Concreta e Gerência da Interface Final.

O primeiro passo sugerido pelo *framework* CAMELEON, Modelar Tarefas e Conceitos, não foi considerado nesta abordagem, uma vez que há abordagens bem compreendidas e amplamente utilizadas para esta etapa, cujo objetivo é descrever as tarefas que o usuário deve desempenhar para alcançar um objetivo quando interage com um sistema computacional [74, 50, 61, 32], conforme citado na Seção 2.1.

Com a modelagem das tarefas e conceitos do domínio realizada, inicia-se o processo de modelagem de IHC proposto neste trabalho, conforme apresentado na Figura 5.1. Para cada tarefa independente, deve haver um estereótipo específico que atenda ao propósito da tarefa. Caso o estereótipo não tenha sido modelado, isto deve ser realizado neste momento.

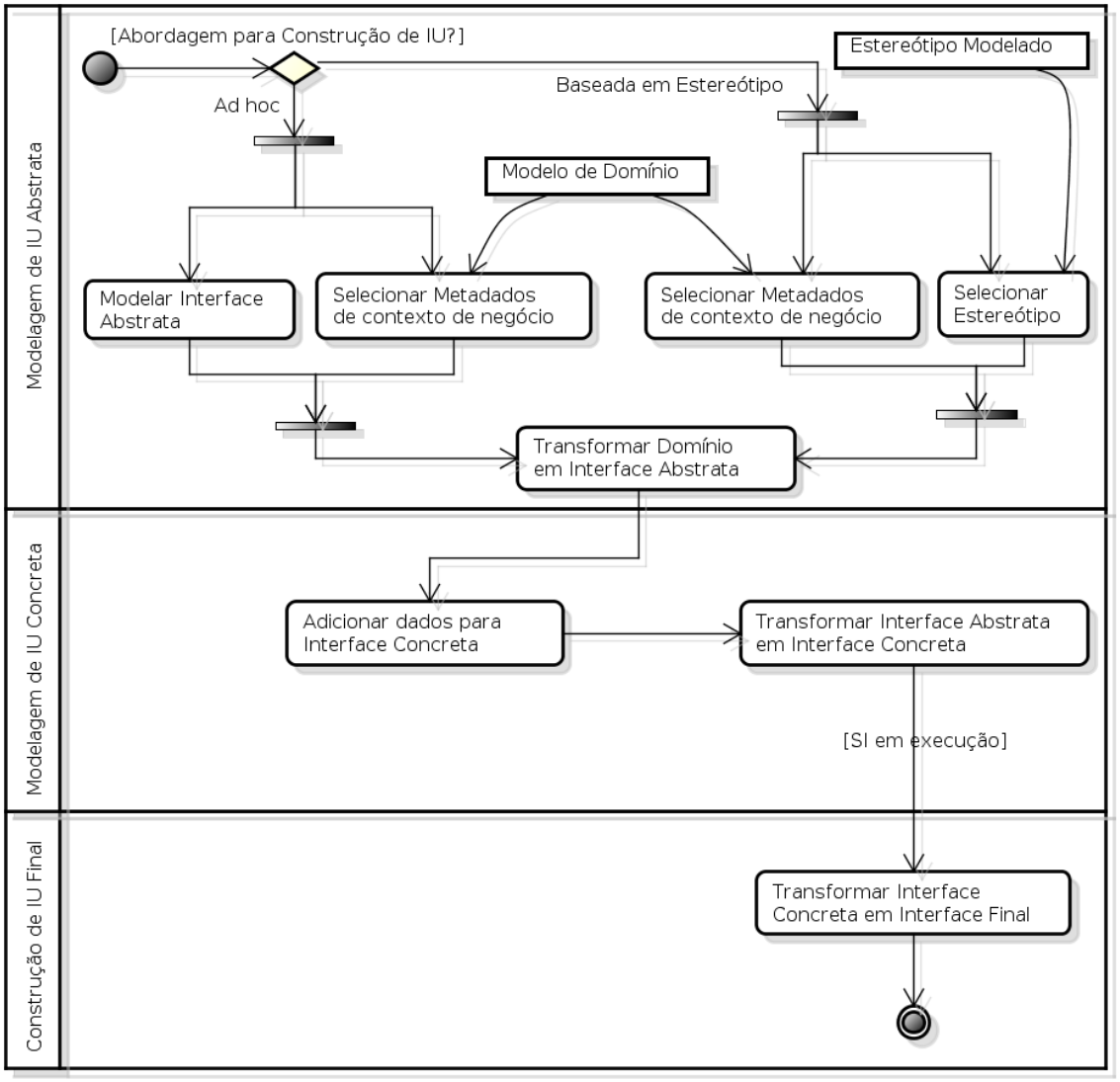


Figura 5.1: *Cenários para geração de IU baseada em modelos*

Para construir interfaces utilizando-se estereótipos de interface, como proposto neste trabalho, é necessário ter uma instância do Metamodelo de Domínio e conter o estereótipo desejado modelado e implementado no componente de Gerência de IU, como visto na Figura 5.1.

Para que o processo de modelagem de IU Abstrata seja realizada (Figura 5.1), deve-se selecionar um estereótipo e associá-lo às informações de elemento de negócio que devem ser apresentadas na interface. Ou seja, em cada instância de Estereótipo há metadados abstratos de interface que serão definidos com base nos metadados de negócio representados através do Metamodelo de Domínio (Seção 3.1). Uma vez que os metadados de negócio são conhecidos, eles podem ser transformados pelo pacote Transformador Domínio - Interface Abstrata, combinando os dados de um elemento do negócio a dados de um estereótipo de interface.

Com os metadados abstratos, é possível realizar o processo de Modelagem de

IU Concreta (Figura 5.1), através da transformação da interface abstrata para interface concreta, ou seja, tendo definido em alto nível de abstração as informações que devem ser apresentadas para o usuário, estas podem ser mapeadas para elementos específicos de uma plataforma automaticamente por meio do componente de Gerência de IU.

No presente trabalho, a plataforma alvo compreende IU WIMP, com a utilização de JSF/Java. Isto deve ser realizado pelo pacote Transformador Interface Abstrata - Interface Concreta.

O modelo de interface concreta é armazenado pelo pacote Metamodelo de Apresentação. A partir deste modelo, as IU são mapeadas para linguagem computacional pelo pacote Interpretador Interface Final durante a execução do Sistema de Informação.

5.2 Modelagem de Estereótipo de Interface

Modelar um Estereótipo de Interface requer a abstração da apresentação e do comportamento da interface que pode ser aplicada em diferentes situações em SI.

Cada Estereótipo identificado no contexto de SI pode ser aplicado para a geração de diferentes instâncias de IU. Por exemplo, o Estereótipo Formulário, identificado na Seção 3.2.3, pode ser instanciado para manipular diferentes elementos de negócio de SI, como para a entidade Pessoa, mostrada na Seção 3.1, ou outros elementos de negócio, como Produto, Nota Fiscal, entre outros.

A abstração do aspecto de apresentação de um estereótipo deve estabelecer um conjunto de elementos de interação que estão presentes em qualquer interface baseada no estereótipo.

Deste modo, haverá alguns elementos fixos em todas as instâncias do estereótipo. No entanto, pode haver espaço para elementos de interação dependentes das informações do domínio. Neste espaço, é aplicado o Transformador Domínio - Interface Abstrata, que realiza o mapeamento dos metadados do elemento de negócio para elementos de interação.

Em relação à abstração do comportamento, cada estereótipo deve ter uma especificação de ações com as funcionalidades associadas aos elementos de interação. Essas funcionalidades podem ser delegadas para o estereótipo em nível de interface, ou podem ser implementadas pela aplicação ou sistema externo.

5.2.1 Estereótipo de Portal Corporativo

Segundo [70], um Portal oferece acesso a uma ampla variedade de informações sobre determinada organização. Nesse sentido, Portais são *Web sites* que permitem ao público-alvo acessar e interagir com informações relevantes, aplicações e processos de

negócio. Este tipo de aplicação é muito empregado para as organizações mostrarem seus produtos, serviços e atividades no contexto global da *Web* [62].

A Figura 5.2 apresenta uma abstração da forma de apresentação de um Portal, adaptada de [70]. Nesta abstração, um Portal é dividido nas seguintes partes:

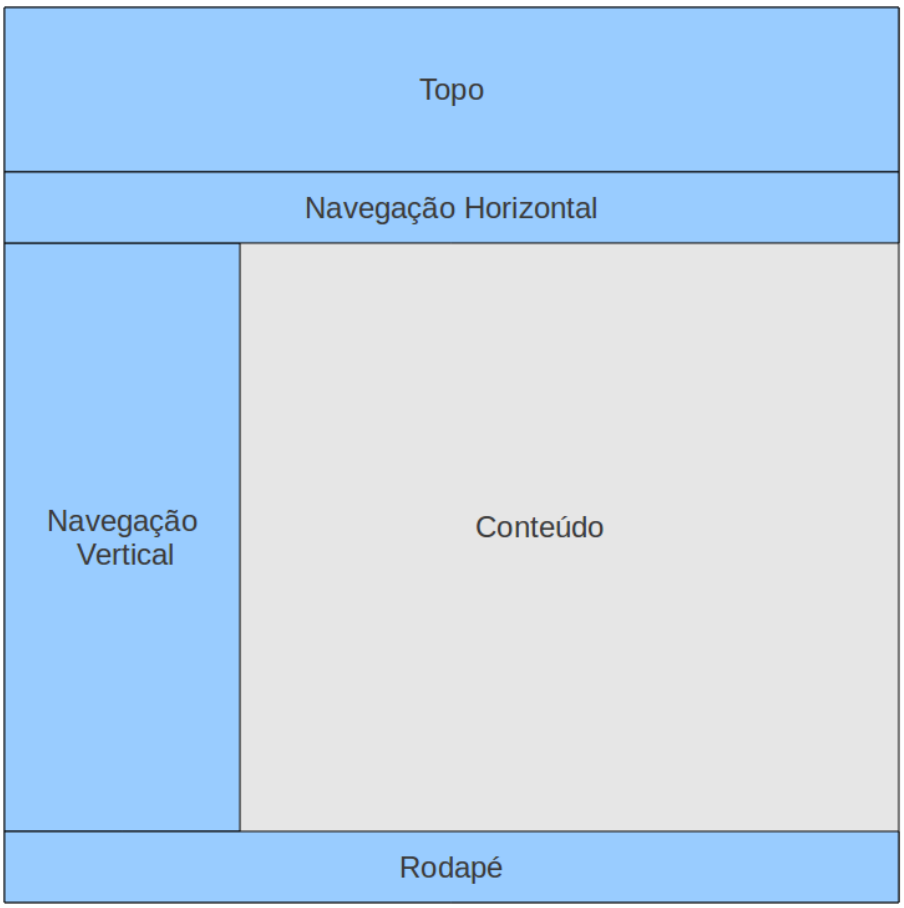


Figura 5.2: *Estereótipo de Interface Portal*

- Topo: apresenta a identificação da organização, com um logotipo ou marca representado por elementos de informação.
- Navegação horizontal: apresenta elementos de informação do tipo serviço, que possibilitam acessar os subsites ou ainda subsistemas que compõem o Sistema de Informação.
- Navegação vertical: exibe elementos de informação do tipo serviço, permitindo a execução de tarefas específicas dos subsistemas, complementando a navegação horizontal por oferecer maiores detalhes das funcionalidades e informações do sistema.
- Conteúdo: parte do estereótipo que possibilita apresentar informações ou agregar outro estereótipo. Como padrão é exibido um estereótipo de página inicial (veja Seção 5.2.2), e com o decorrer da navegação esta parte é atualizada com os estereótipos solicitados.

- Rodapé: contém itens informativos a respeito do sistema desenvolvido e da organização, como nome, endereço e formas de contato, entre outras informações.

A Figura 5.3 apresenta o modelo de IHC para o estereótipo Portal. Um portal deve conter o topo como um agrupador, navegação horizontal e vertical como agrupadores de itens de menu, que são do tipo informação de serviço. Cada item de menu deve disparar uma ação interna do estereótipo cuja responsabilidade é atualizar a interface de acordo com a solicitação realizada pelo usuário por meio do menu. Na parte de conteúdo, não há ações predefinidas, a não ser que seja agregado outro estereótipo que contenha outras ações. O rodapé apenas apresenta informações, não permitindo interação com elas.

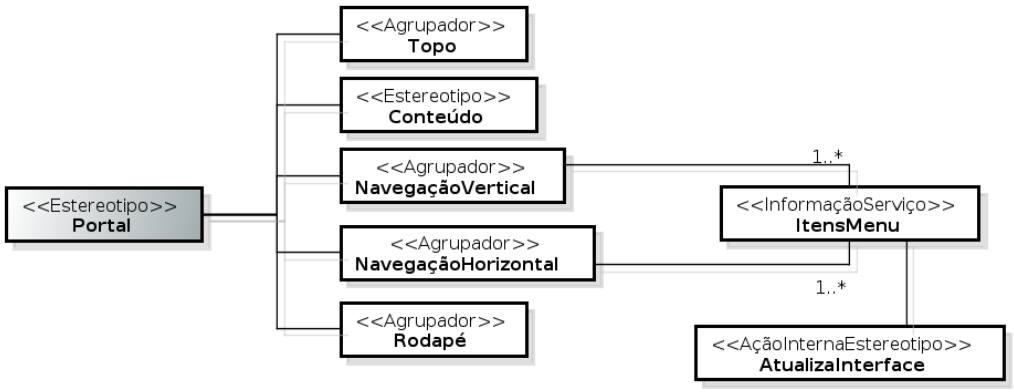


Figura 5.3: *Modelagem do Estereótipo de Interface Portal*

Neste estereótipo, o tipo de ação requerida nas partes de navegação horizontal e vertical é de navegação. Os itens de serviço presentes nas partes de navegação horizontal e vertical devem disparar uma ação de interface, que não necessita ser implementada pela aplicação. A ação recebe como parâmetro o nome do item de serviço que será utilizado para direcionar a interface para outro estereótipo ou recarregar a interface.

A Figura 5.4 exibe a implementação do estereótipo Portal como um *Template* no Metamodelo de Apresentação. É implementada a classe TemplatePortal, possuindo os respectivos elementos de interação e a ação de interface implementada no estereótipo. Esta classe ainda deve ser identificada como um *ManagedBean*, para que suas regras sejam acessadas pela interface correspondente.

Como o conteúdo do Portal é composto por outros estereótipos, ele deve conhecer o estado corrente de cada um desses estereótipos. Isto quer dizer que, se o estereótipo em execução disparar uma regra de estereótipo que conduza a carregar outro estereótipo em seu lugar, o estereótipo Portal deve conhecer este fato para carregar a interface solicitada adequadamente.

Além da classe TemplatePortal, deve haver um arquivo JSF com a descrição da aparência da interface final. Neste arquivo devem ser incluídos os componentes paineis, como apresentadas na Seção 4.3.2, para compor o template. Para as partes de navegação

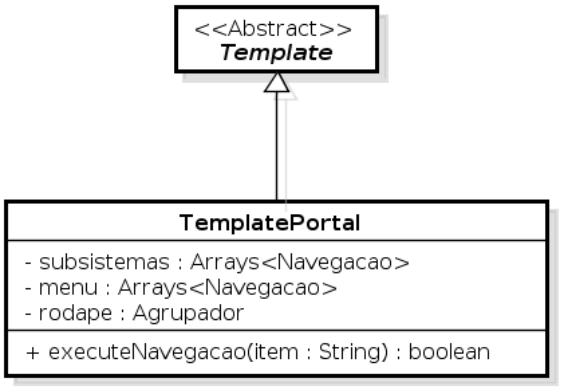


Figura 5.4: *Fachada de regras para Portal*

horizontal e vertical haverá um algoritmo em JSF que, para cada item de navegação, acrescentará os componentes de navegação para a instância do template.

5.2.2 Estereótipo de Conteúdo de Página Inicial

Este estereótipo deve ser agregado a um Portal, sendo a parte interna que contém as informações que o usuário deve acessar por meio do Portal. Ele pode reunir informações que introduzam o site ao usuário, mostrando quais as funcionalidades e informações disponíveis acerca da organização. Geralmente isto é exibido em formato de notícias, tendo algumas de maior destaque. É o ponto de partida da interação do usuário com a aplicação.

Uma possível forma de apresentação, dentre inúmeras possíveis, é ilustrada na Figura 5.5. Seus elementos são os seguintes:

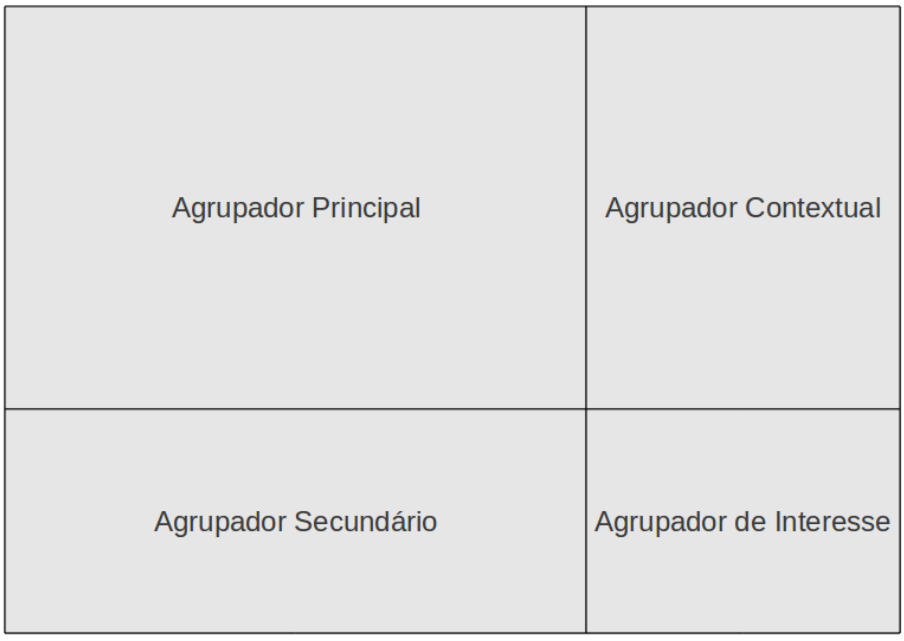


Figura 5.5: *Estereótipo de Interface Conteúdo de Página Inicial*

- Agrupador Principal: apresenta elementos de destaque da página representados por elementos de informação do tipo textual ou visual.
- Agrupador Secundário: apresenta elementos de menos destaque, mas que trazem informações importantes, como notícias. São utilizados elementos de informação textual para preenchê-lo.
- Agrupador Contextual: parte do estereótipo que pode conter informações contextuais do tipo visual ou textual, ou ainda conter serviços disponíveis no *site*.
- Agrupador de Interesse: traz um formulário para que o usuário preencha, caso tenha interesse em notícias ou outra informação da organização.

Do mesmo modo que no estereótipo Portal, os dados concretos deste estereótipos necessitam da implementação da classe TemplateConteudoPag, exibida na Figura 5.6, que implementa a classe Template com as propriedades agrupadorPrincipal, agrupadorSecundario, agrupadorContextual e agrupadorInteresse. Estas propriedades reúnem os elementos de cada agrupador deste estereótipo.

A ação de interface é do tipo navegação, e é predefinida via estereótipo, denominada executeNavegacao. Esta ação recebe como parâmetro um item do estereótipo que identifica a transição de interface que será executada. Além disso, um arquivo em JSF deve conter os componentes paineis que fazem parte da interface de forma predefinida, e o mecanismo que preenche os componentes que variam em cada instância do template.

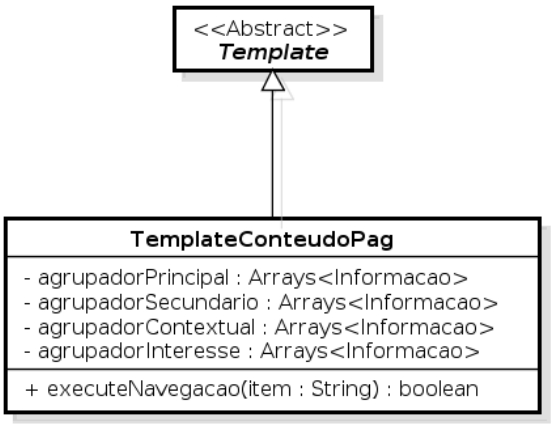


Figura 5.6: *Fachada do Estereótipo de Interface Conteúdo de Página Inicial*

5.2.3 Estereótipo CRUD

O Estereótipo CRUD (*Create, Retrieve, Update, Delete*) é muito comum em aplicações que envolvem a manipulação de informações armazenadas em bancos de dados, sendo uma característica típica de SI.

A Figura 5.7 apresenta o modelo de IHC para o estereótipo CRUD. Uma instância de CRUD é composta por dois outros estereótipos: Formulário e Planilha. O estereótipo Formulário, apresentado na Seção 3.2.3, tem como objetivo a entrada de dados no sistema. Este estereótipo permite apresentar apenas elementos de negócio do tipo entidade, entidade fraca ou relacionamento.

O formato de apresentação do Formulário é exibido na Figura 3.4. Formulário deve ter um título, um agrupador de campos de entrada, que deve ser preenchido com questões, conforme o domínio de informações, e um agrupador de funções do formulário.

As funções de CRUD já são conhecidas, e são especificadas pelo estereótipo por elementos de informação do tipo serviço: Salvar, Limpar e Cancelar.

Salvar deve disparar uma ação implementada externamente ao componente de IU. Limpar e Cancelar disparam ações de interface, sendo limpar para deixar em branco os campos de entrada do formulário, e cancelar para desabilitar o formulário.

O estereótipo Planilha tem como finalidade apresentar uma tabela ordenada com informações de determinado elemento de negócio. Por meio da planilha estas informações podem ser não apenas visualizadas, mas também manipuladas. Ela deve possuir uma tabela, onde cada coluna representa um atributo de um elemento de negócio e uma linha representa uma instância desse elemento de negócio.

Deve-se ressaltar que a Planilha é o único estereótipo identificado neste trabalho que permite representar um elemento de negócio do tipo consulta, uma vez que por restrições que não fazem parte do escopo deste trabalho, ainda não é possível realizar a manipulação (inserção e edição) de um elemento deste tipo, sendo utilizada apenas para consultar informações do negócio.

Para evitar a navegação horizontal, nem todos os atributos do elemento de negócio devem aparecer na planilha. Através da interface do componente de gerência de IU, devem ser selecionados quais atributos devem aparecer na planilha.

O estereótipo Planilha é composto por uma tabela, que deve conter linhas suficientes para a quantidade de registros do elemento de negócio, e um elemento de informação do tipo serviço inserir. Este serviço deve disparar uma ação que é responsável por habilitar o formulário para inserir um novo registro ao elemento de negócio.

Devem existir três ações para cada linha da tabela: editar, carregando no formulário os dados do negócio e disponibilizando-o para alteração; excluir, fazendo com que apareça uma mensagem de confirmação de exclusão ao ser acionada esta regra; e consultar, que permite visualizar todos os atributos do elemento de negócio.

Acima da planilha deve haver um elemento de serviço que dispare a regra inserir, que irá disponibilizar o formulário em branco para entrada de uma nova instância do elemento de negócio.

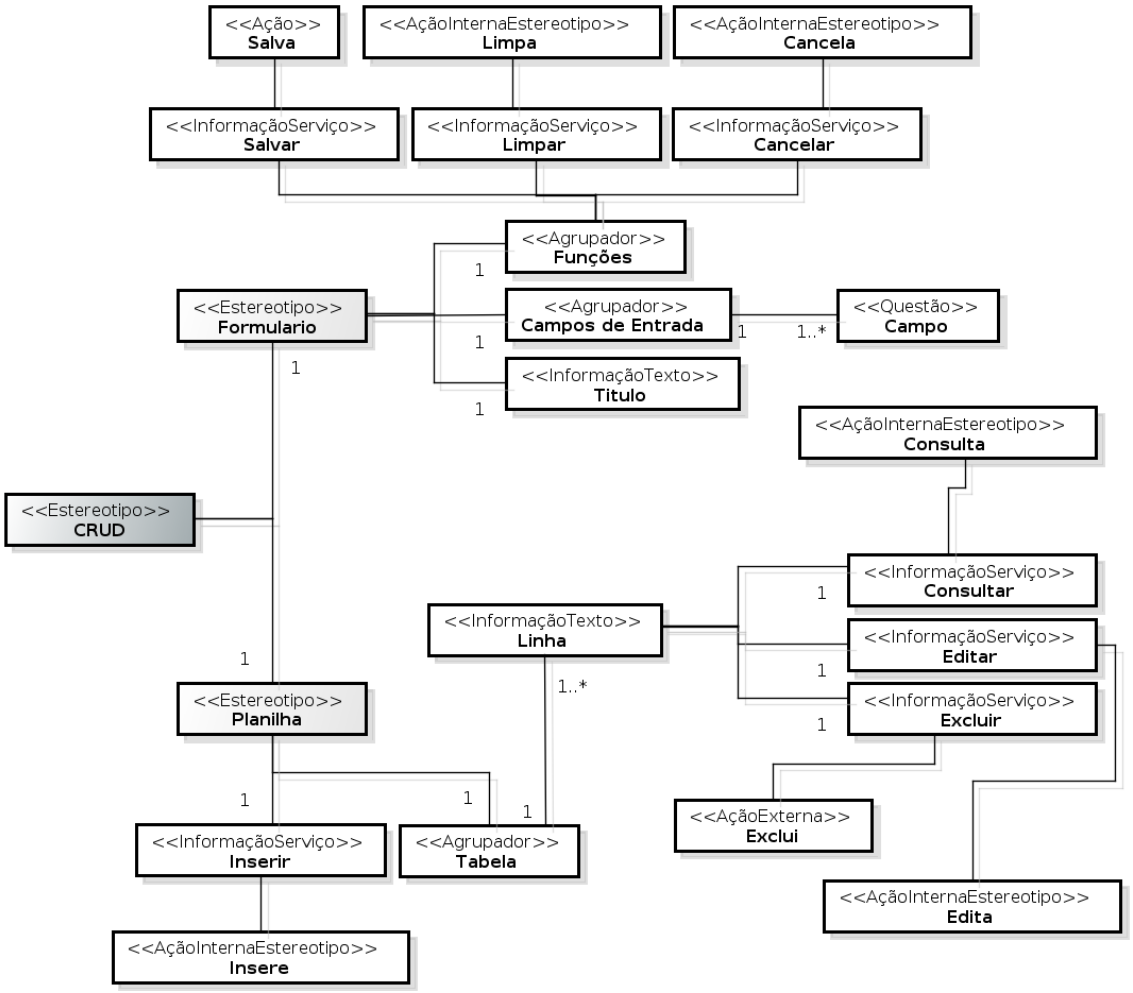


Figura 5.7: *Modelagem do Estereótipo de Interface CRUD*

A Figura 5.8 exibe a composição dos dois estereótipos - Formulário acima e Planilha abaixo - formando o estereótipo CRUD.

Os dados concretos envolvem as regras comportamentais a serem disparadas, que são disponibilizadas por meio de uma API de regras, para que sejam implementadas. A Figura 5.9 exibe como implementar este estereótipo. Deve ser implementada a classe Template no pacote Metamodelo de Apresentação, contendo como atributo um objeto de Domínio, que recuperará os dados do domínio a serem manipulados na interface, e armazenará as entradas do usuário para que seja devolvido à aplicação. Também deve referenciar o estereótipo na qual a instância deste estereótipo deve ser agregado. A operação *execute* será responsável por transferir para a aplicação a execução da regra que foi disparada. A aplicação irá basear-se na *Interface* que define as funcionalidades esperadas pela interface.

A Figura 5.9 exibe a classe TemplateCrud que deve ser implementada. Esta classe deve implementar a fachada para os métodos limpar e cancelar. Também deve ser implementada uma fachada CRUD na Regra de Aplicação, que manipula as regras

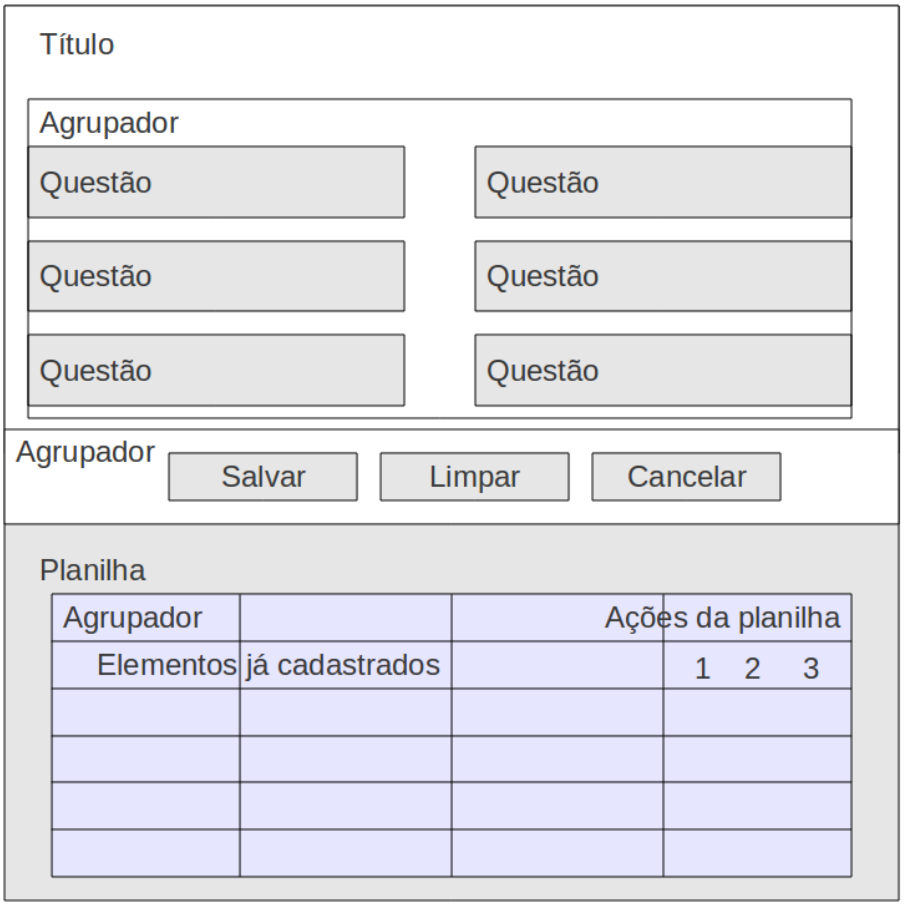


Figura 5.8: *Estereótipo de Interface CRUD*

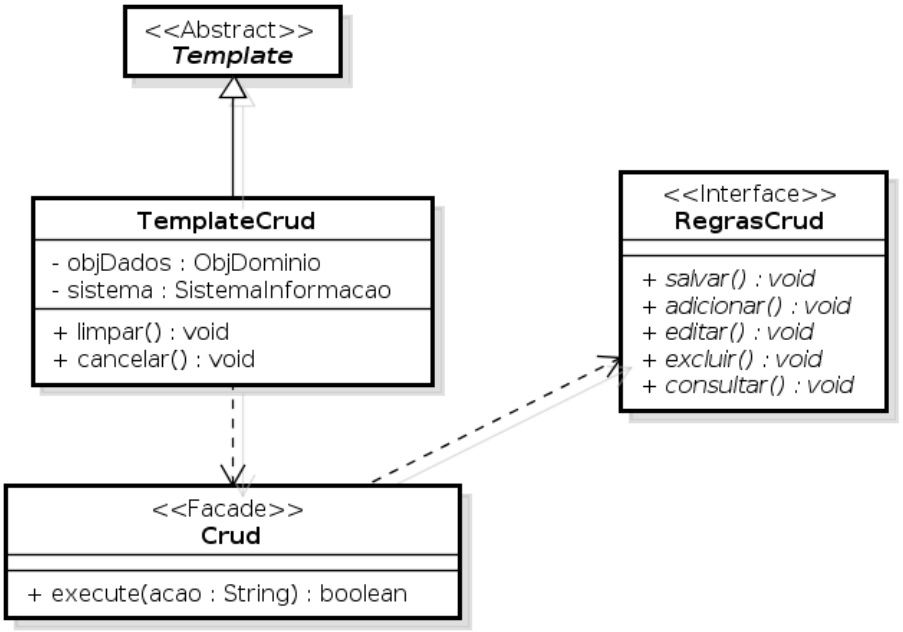


Figura 5.9: *Fachada de regras para Interfaces CRUD*

definidas na *interface* CRUD (salvar, adicionar, editar, excluir e consultar).

Além disso, deve haver um arquivo JSF com a descrição dos painéis utilizados

pelo template, bem como o algoritmo que acrescenta os painéis específicos para uma instância do template. A Figura 5.10 mostra um exemplo de template de formulário em JSF, onde o bloco *forEach* é o algoritmo que adiciona os painéis conforme cada instância do template. O bloco *ui:include* inclui um painel de navegação do tipo botão.

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
                xmlns:h="http://java.sun.com/jsf/html"
                xmlns:ui="http://java.sun.com/jsf/facelets"
                xmlns:c="http://java.sun.com/jsp/jstl/core"
                xmlns:o="http://openfaces.org/">

 <h:form id="form">


     <c:forEach items="#{template.paineis}" var="p" >
        <ui:include src="#{p.html}">
            <ui:param name="p" value="#{p}"></ui:param>
        </ui:include>
     </c:forEach>


     <ui:include src="/paineis/painelNavegacaoBot.xhtml">
        <ui:param name="b" value="#{template.botao}"></ui:param>
        <ui:param name="t" value="#{template}"></ui:param>
     </ui:include>

 </h:form>

</ui:composition>
```

Figura 5.10: *Template em JSF para Estereótipo Formulário*

5.3 Transformação de Domínio em Interface Abstrata

Com o estereótipo definido, antes de concluir a geração da interface abstrata é necessário associar a interface a um elemento de negócio, caso o estereótipo necessite disto. Então os metadados de domínio do negócio serão considerados para que seja gerada uma interface adequada a estes metadados.

Através do componente para gerência de IU, descrito no Capítulo 4, deve-se selecionar um estereótipo de interface e o elemento de negócio a ser apresentado pela interface a ser gerada. As regras de transformação para o pacote Transformador Domínio - Interface Abstrata para o estereótipo Formulário são mostradas no Apêndice A.1.

Deste modo, para uma interface que emprega um estereótipo do tipo Portal com um estereótipo CRUD e um elemento de negócio do tipo Pessoa (como mostrado no exemplo da Seção 3.2.4), tem as seguintes informações como entrada para o transformador:

Pessoa: entidade que tem os atributos Nome (alfanumérico), Data de Nascimento (data), CPF (numérico), Idade (numérico) e Naturalidade composta de Estado (enumerado), Cidade (enumerado).

Estereótipo CRUD: elementos de informação do tipo serviço do formulário, elemento agrupador (tabela) com elementos de informação do tipo serviço na última coluna da planilha e elemento de informação do tipo serviço para inserir novos dados na planilha.

O estereótipo de interface já possui uma descrição abstrata predefinida, e este deve ser instanciado para que possam ser acrescentados os dados abstratos referentes ao domínio do negócio.

A Figura 5.11 exibe o resultado das transformações realizadas pelo pacote Transformador Domínio - Interface Abstrata, gerando o modelo abstrato para a interface CRUD de Pessoa. No mapeamento do domínio para o formulário, cada atributo do elemento de negócio Pessoa deve ser mapeado para um elemento de entrada, gerando os dados abstratos do formulário, como mostra a Figura 3.5.

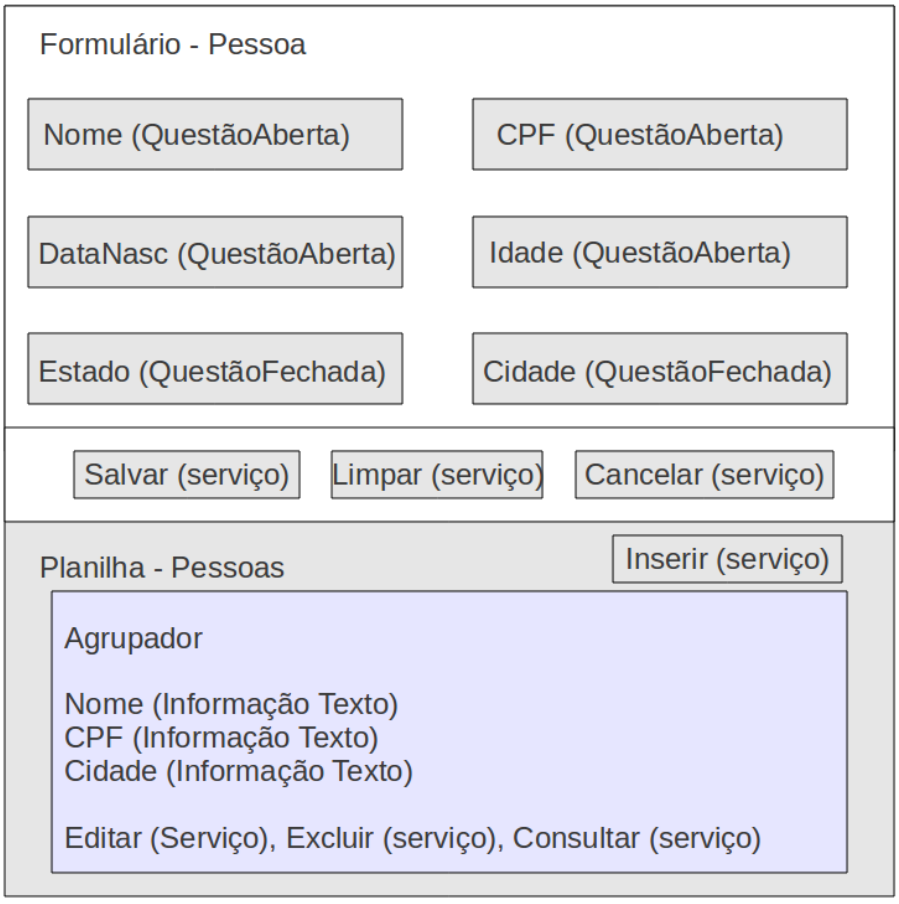


Figura 5.11: *Interface Abstrata gerada para Pessoa*

Para mapear a entidade para a planilha, cada atributo deve ser associado a uma coluna da tabela, e as linhas devem ser preenchidas, em tempo de execução, com os registros de Pessoa. A última coluna deve ser preenchida com os elementos de informação do tipo serviço, para que possam ser disparadas as ações para cada registro de Pessoa.

Já para o estereótipo Portal, esta etapa envolve a associação dos dados do sistema como um todo, ou seja, as ligações que conduzirão a cada estereótipo específico. Deste modo, para um contexto de organização encontrado na *Web* (http://www.estrategia.eti.br), como apresentado na Figura 5.13(a), foram identificados os dados de domínio (Figura 5.12). Os elementos de negócio (com seus respectivos atributos) identificados para este portal são Subsistemas (Consultoria, Treinamento, Diagnóstico, Clientes e Suporte), Conteúdo (Missão, Notícias, Frase do Dia e *Newsletter*) e Acesso Restrito (Usuário e Senha). Os atributos de Subsistemas são considerados alfanuméricos porque possuem um *link* associado a eles.

A partir dos dados do domínio, foram gerados os metadados abstratos, como pode ser observado na Figura 5.13(b).

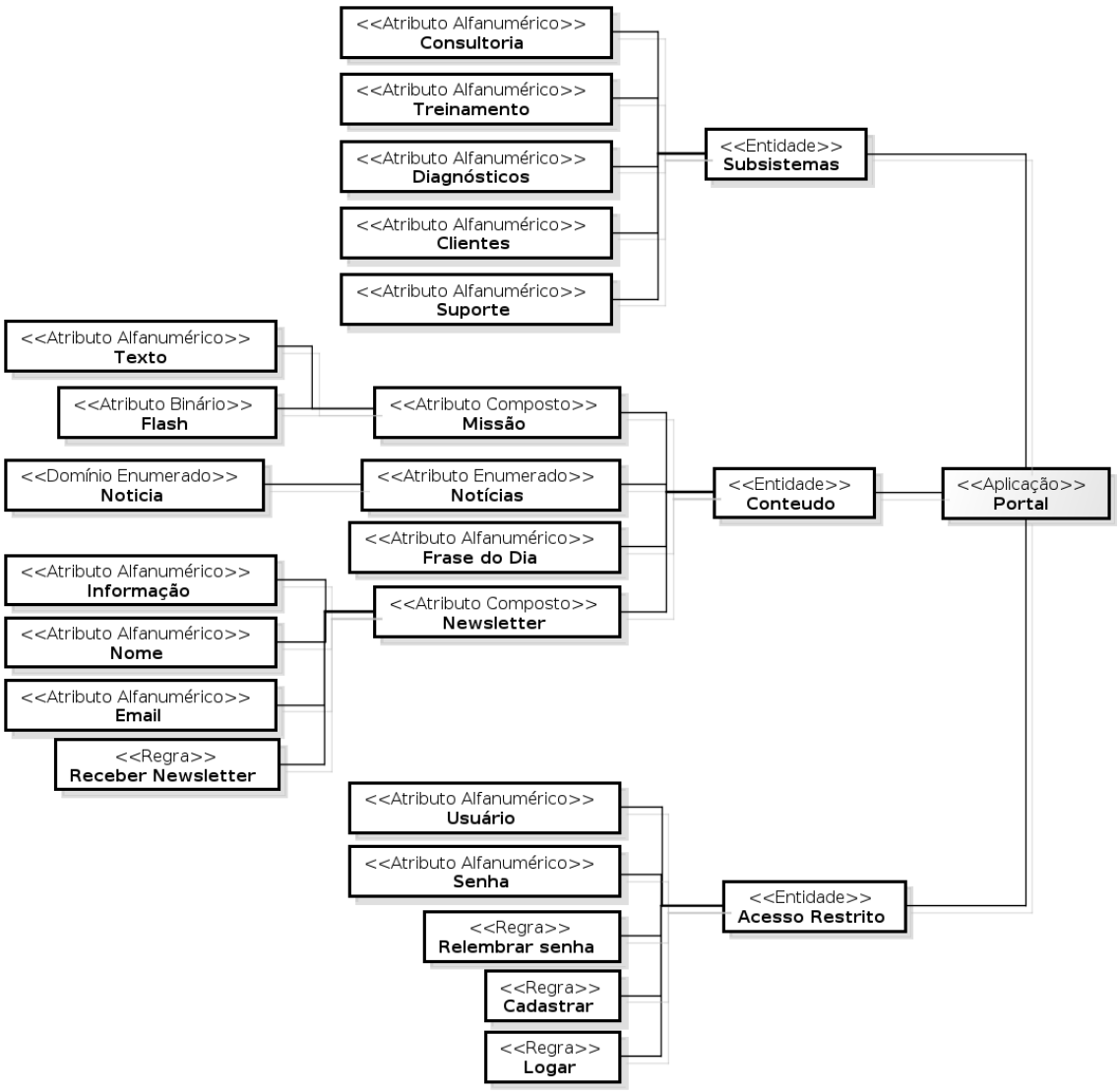
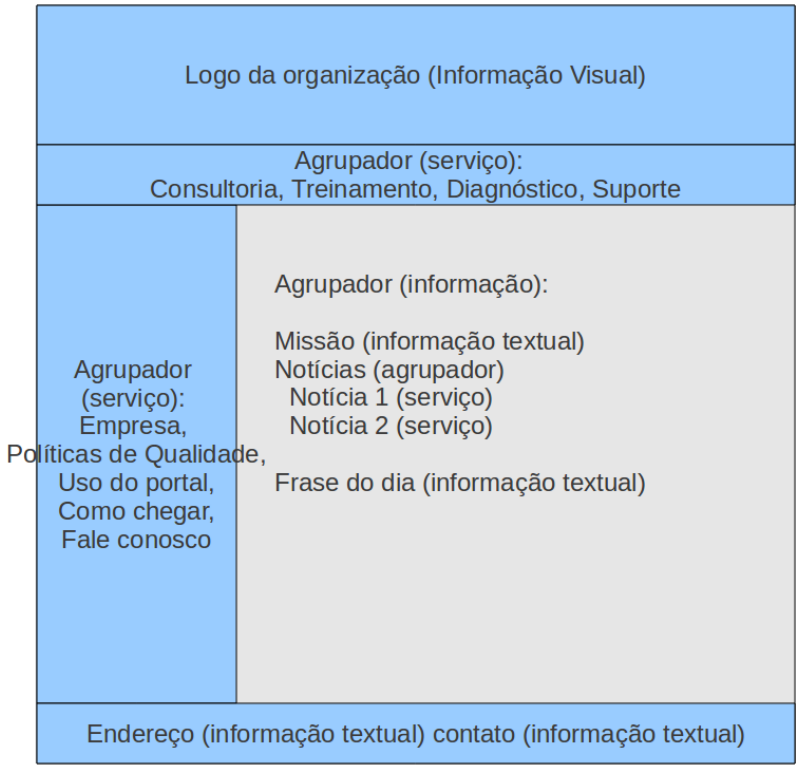


Figura 5.12: *Modelagem de Domínio do Portal de uma empresa*

(a) *Site com Portal de uma empresa*



(b) *Interface Abstrata gerada para Portal de uma empresa*

Figura 5.13: *Portal de uma empresa*

5.4 Transformação de Interface Abstrata em Interface Concreta

Com base nos dados da interface abstrata definida, pode-se gerar a interface concreta com o pacote Transformador Interface Abstrata - Interface Concreta. O estereótipo CRUD já deve conter uma especificação concreta predefinida para os elementos fixos. Os elementos que foram acrescentados, referentes ao domínio do negócio, precisam ser mapeados para uma descrição concreta.

Para esta transformação, são consideradas as regras de mapeamento apresentadas no Apêndice A.2. Deste modo, a interface concreta gerada para estereótipo CRUD do elemento de negócio Pessoa é exibida na Figura 5.14.

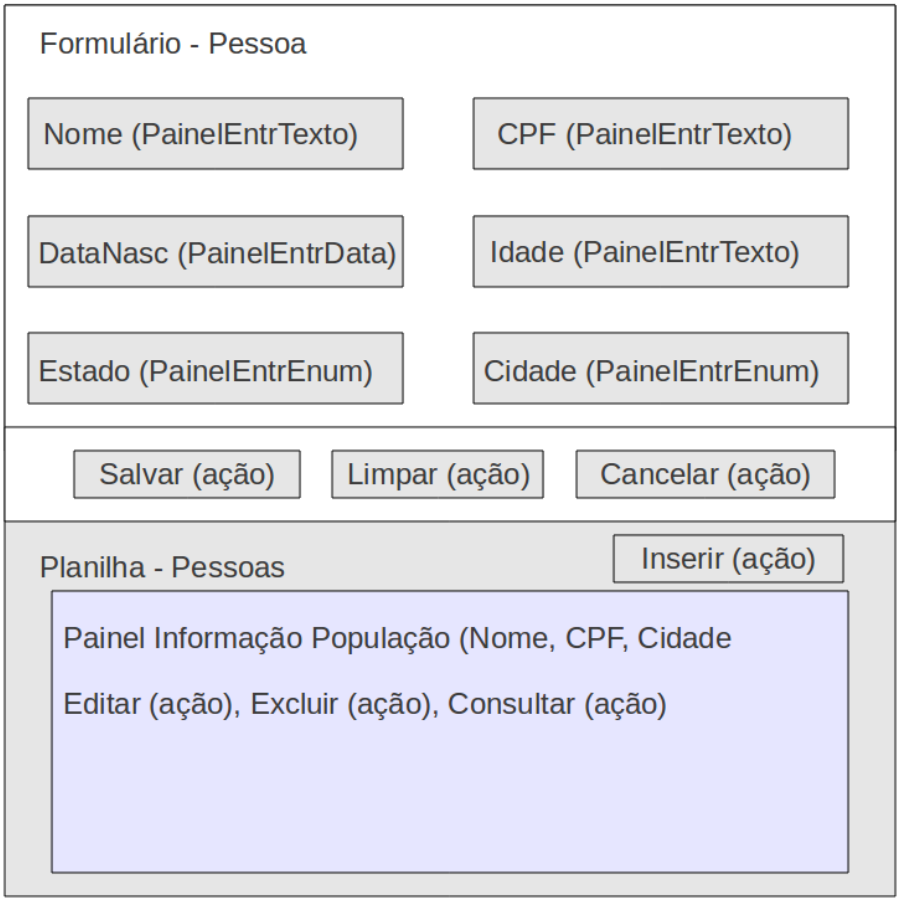


Figura 5.14: *Interface Concreta gerada para Pessoa*

Para o estereótipo Portal, os dados concretos gerados são vistos na Figura 5.15. Os dados concretos são associados a componentes JSF, logo possuem algumas propriedades específicas desta plataforma, como referência a classe CSS que representa o estilo do painel e a posição em que o painel se encontra no leiaute do template, além da implementação do mecanismo de regras via API. Os elementos de serviço foram

mapeados para paineis de navegação do tipo menu, as notícias para *links* e as informações textuais para paineis textuais.

A interface concreta gerada pode ser customizada para seguir preferências de usuário ou melhorar a usabilidade da interface, já que existem diferentes tipos de paineis para um mesmo tipo de elemento de interação do metamodelo abstrato, conforme os mapeamentos do Apêndice A.2.

Os dados concretos devem ser armazenados em banco de dados pelo pacote Metamodelo de Apresentação, para que no momento da execução da aplicação possam ser interpretados para compor a interface final.

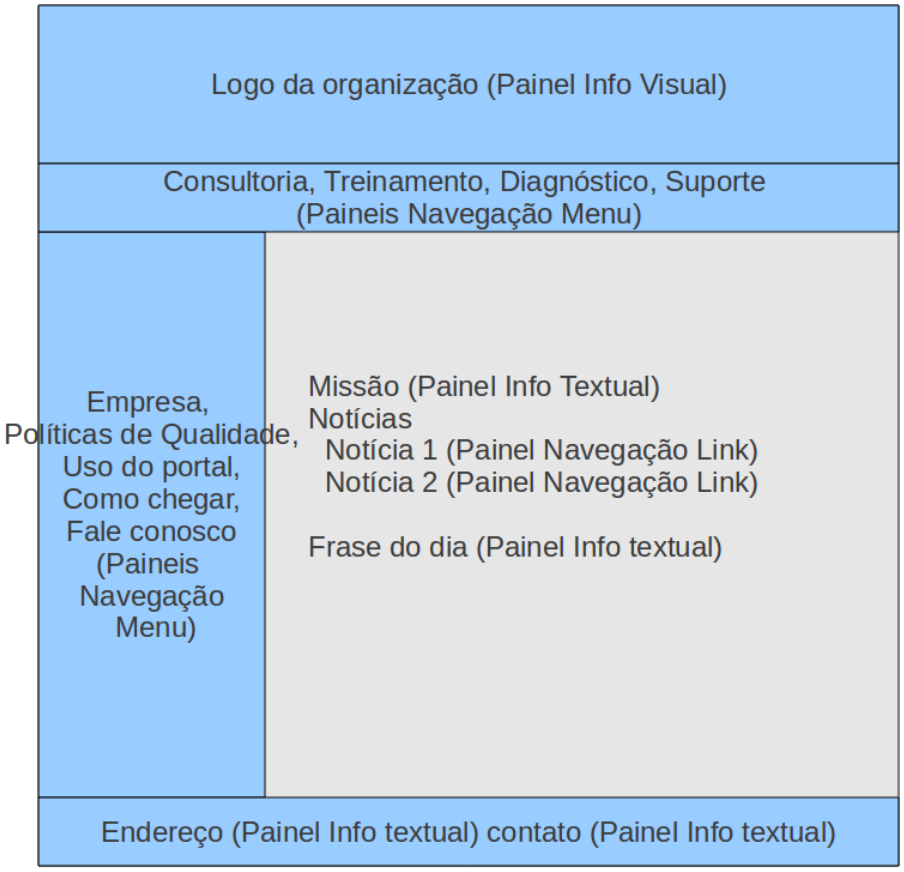


Figura 5.15: *Interface Concreta gerada para Portal de uma empresa*

A fachada de regras do template Portal (Figura 5.4) serve como um contrato entre a interface e a aplicação ou o estereótipo, descrevendo quais regras devem ser implementadas para que a interface funcione corretamente.

5.5 Interpretação da Interface Concreta para Execução

Esta atividade ocorre no momento em que o usuário solicita a execução de uma interface. A Figura 5.16 mostra como ocorre a geração das interfaces e a gerência de

seu comportamento durante a execução do Sistema de Informação em um Diagrama de Estados.

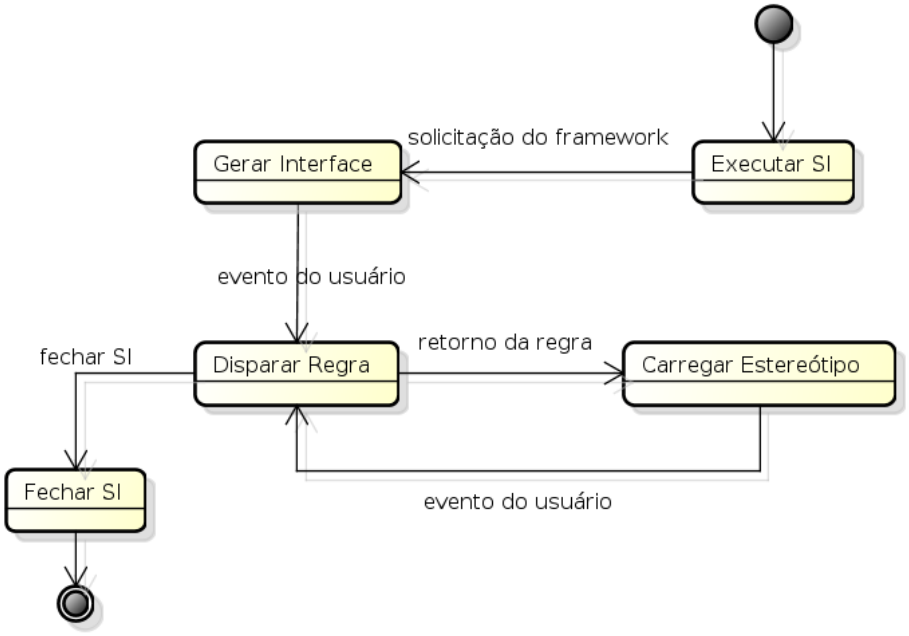


Figura 5.16: *Ciclo de vida de IU em SI*

Uma vez que o sistema entra em execução por meio da solicitação do usuário, o *framework* de gestão de SI faz uma requisição ao componente de gerência de IU para executar a interface final do Estereótipo que é identificado como o estereótipo padrão para iniciar o sistema para o usuário. Ou seja, o estado Executar SI é implementado pelo *framework* de gestão de SI. Este *framework* faz uma solicitação ao componente de Gerência de IU para que a interface padrão seja gerada e apresentada ao usuário. Assim, o estado Gerar Interface é implementado pelo pacote Interpretador Interface Final, que retorna a interface padrão do sistema.

A interface apresentada ao usuário aguarda um evento de usuário nos componentes de navegação, e estes devem disparar a respectiva regra de interface. Assim que o usuário realiza uma interação, é disparada uma regra via classe Template do pacote *ManagedBeans* presente em Interpretador Interface Final.

Assim que a regra disparada retorne uma resposta, o sistema é conduzido ao estado Carregar Estereótipo, que também é implementado pela classe Template do pacote *ManagedBEans*. Este estado atualizará a interface com as informações retornadas pela regra. Isto se repetirá cada vez que ocorrer um evento de usuário, passando do estado de Carregar Estereótipo para Disparar Regra novamente.

Se a regra disparada for uma regra para fechar o sistema, o estado Fechar SI irá disparar as regras necessárias para que o sistema seja fechado com segurança. Isto é,

a classe Template disparaá uma regra para a aplicação fechar o sistema e a interface é fechada.

Para a geração de cada interface, os dados concretos devem ser interpretados de forma que sejam carregados os componentes gráficos correspondentes a cada painel. Para cada estereótipo contido na interface, são seguidos dois passos para a geração. Primeiro são gerados os componentes correspondentes a cada painel predefinido no estereótipo; segundo, para cada painel específico do elemento de negócio, gerar o painel correspondente da biblioteca de componentes.

5.6 Integração da Abordagem com a Construção de SI

Sistemas de Informação (SI) são construídos de modo a apoiar a troca de informações e provendo o conhecimento necessário para a tomada de decisão sobre o negócio de uma organização [63].

Construir SI envolve, no mínimo, o projeto de três componentes principais para gerenciar os aspectos de dados, processos e interação do usuário com as informações do negócio. A maior parte das abordagens para construção automática de interfaces não trata da integração destes aspectos no processo de desenvolvimento de SI.

A integração entre os aspectos de Domínio de Negócio e Interface de Usuário, introduzida em [15], pode ser observada na Figura 5.17. Nesta figura, aparecem apenas os conceitos principais de cada metamodelo, para exibir como esses conceitos se relacionam.

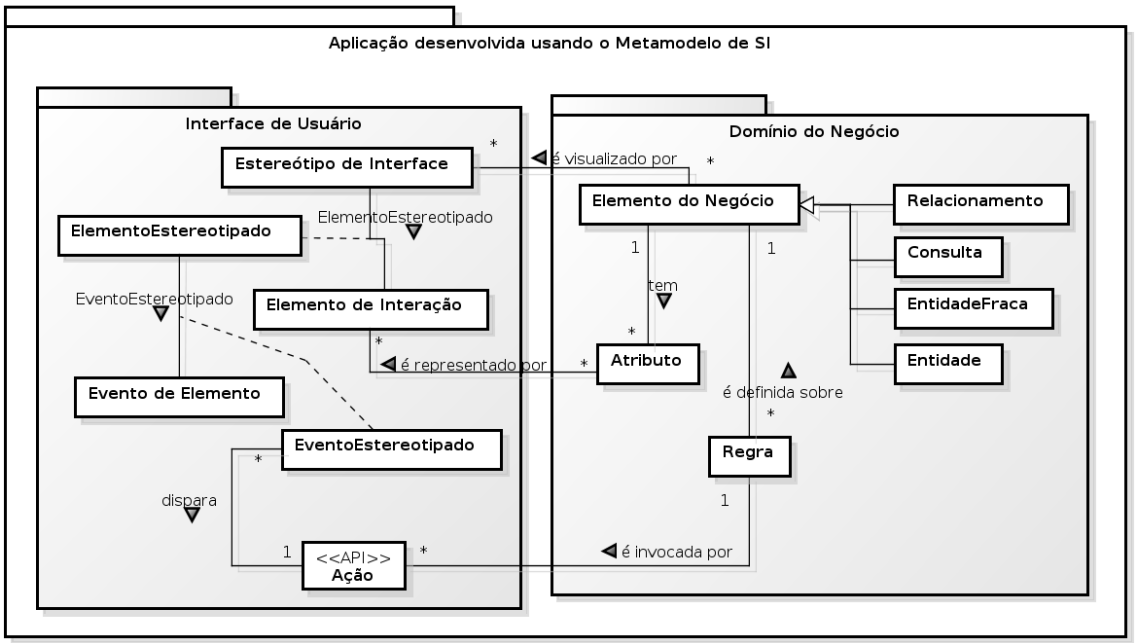


Figura 5.17: *Integração entre os aspectos de Domínio de Negócio e Interface de Usuário*

Com a utilização do Metamodelo de Domínio de Negócio e do Metamodelo de IHC, os metadados são desacoplados e tratados separadamente. O conceito de Estereótipo de Interface possibilita a modelagem de variados tipos de interface, e não apenas CRUD.

Os metadados de negócio podem ser visualizados através de diferentes estereótipos, já que cada estereótipo possui diferentes finalidades de apresentação de informação. Analogamente, cada atributo pode ser representado por um diferente Elemento de Interação em cada estereótipo, conforme o objetivo do estereótipo. A relação entre tipos de atributo e os tipos de elementos de interação é definida no pacote Transformador Domínio - Interface Abstrata, onde os metadados de negócio são mapeados para elementos de interação.

A relação dos aspectos de Processo de Negócio com o de Interface de Usuário e de Domínio de Negócio, observada na Figura 5.18, foi adaptada de [34], onde é descrito o Metamodelo de Processo. A modelagem de processos não faz parte do escopo desta dissertação, mas a integração da interface desses processos é de interesse do presente trabalho.

Um processo de negócio pode ser visualizado por diferentes estereótipos específicos, para definir, executar e acompanhar a execução de processos, por exemplo.

Em um processo de negócio, tarefas associadas a papeis geram responsabilidades. Estas responsabilidades, associadas a um estereótipo específico, introduzem o conceito de **autorização**. Este conceito está associado ao controle de acesso de cada tarefa, e isto envolve outro aspecto, que é Segurança de SI. Este aspecto possui particularidades que devem ser tratadas separadamente, mas também de forma integrada às IU.

Quanto ao comportamento do processo de negócio, este tem um conjunto de regras que são especificadas nas conecções entre os objetos de fluxo. Estas regras podem ser disparadas por eventos de usuário, através de regras comportamentais da interface.

O ObjetoDados que é manipulado pelo aspecto de processo pode ser um Objeto de Domínio, que envolve os dados do domínio do negócio a serem manipulados através do processo de negócio e apresentados na interface.

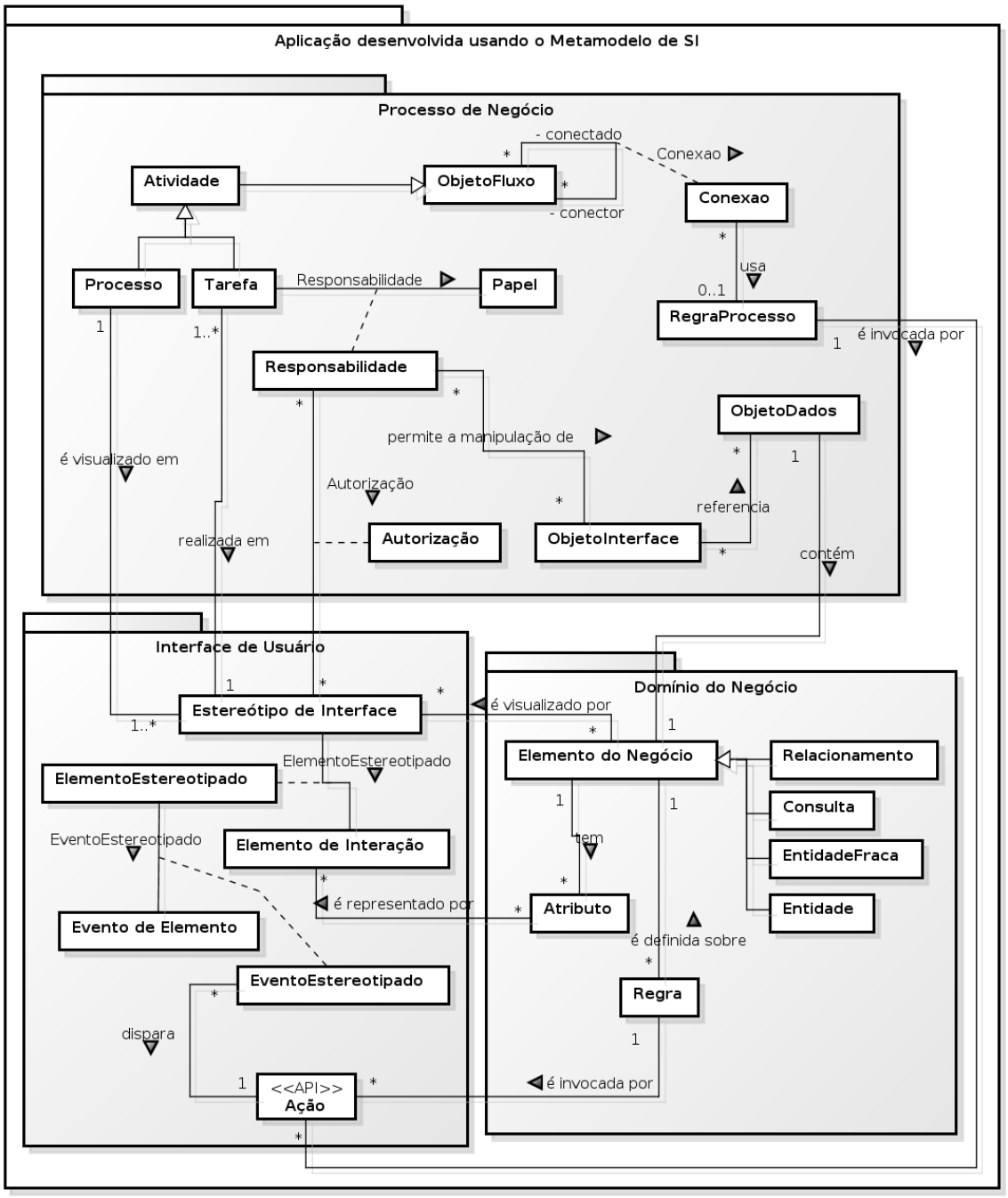


Figura 5.18: *Integração entre os aspectos de Processo de Negócio com Domínio de Negócio e Interface de Usuário*

CAPÍTULO 6

Conclusões

Este trabalho investigou desafios e soluções relacionados à construção e manutenção de IU baseada em modelos para SI. As propostas deste trabalho se apoiam no *framework* para gestão de SI desenvolvido pelo Grupo de Pesquisa do INF, que teve sua primeira versão liberada em 2009 [2], e atualmente encontra-se em evolução, já que algumas dificuldades foram encontradas para a utilização deste *framework* [25].

A Seção 6.1 apresenta as considerações finais deste trabalho. A Seção 6.2 analisa as principais constribuições obtidas com o desenvolvimento desta pesquisa. A Seção 6.3 discute as limitações do trabalho realizado, e a Seção 6.4 indica trabalhos futuros que podem dar extensão a esta pesquisa.

6.1 Considerações Finais

Para que o desenvolvimento de SI seja realizado de maneira eficiente e com qualidade, as pesquisas na área de SI estão se distanciando do enfoque puramente computacional e indo em direção ao seu impacto em pessoas e organizações. Da mesma forma que, na prática, os desenvolvedores de software são focados na tecnologia, os analistas de sistemas devem focar nos conceitos dos negócios subjacentes aos SI.

Assim, há a necessidade de desenvolver SI levando em consideração os pontos de vista tecnológico e humano. Uma vez que o desenvolvimento baseado em modelos é uma abordagem para apoiar a automatização de questões tecnológicas de aplicações, haverá mais recursos a serem alocados para o ponto de vista humano.

Este trabalho apresentou uma abordagem para a construção de IU baseada em modelos, propondo um conjunto de metamodelos empregados no projeto de um componente para geração de IU. Deste modo, algumas das dificuldades citadas em [25] foram superadas, como o desacoplamento dos metadados de domínio de negócio e IU, a geração de IU com diferentes propósitos, e a flexibilidade para mudanças no projeto da interface.

Este trabalho tem sua fundamentação teórica baseada na Engenharia de IU e nos requisitos identificados na Seção 2.3, focando principalmente nos aspectos de

apresentação e comportamento de IU integrado às funcionalidades do SI. Esta pesquisa mostra que:

- A utilização de Estereótipos de Interface beneficia o desenvolvimento de IU do ponto de vista de reutilização, produtividade e consistência. O projeto aqui apresentado fortalece estudos anteriores que mostraram a importância de um elemento que domine e organize a interface de usuário [27, 35, 8, 76], não apenas no que diz respeito aos aspectos de apresentação, como também nos aspectos comportamentais da IU em relação às funcionalidades oferecidas pelo SI.
- O desenvolvimento de IU para SI deve levar em consideração todo o processo de desenvolvimento de SI, e todos os aspectos envolvidos. Uma vez que IU consistem na forma de comunicação entre usuários e SI, elas devem proporcionar uma forma de integrar-se a estes aspectos de forma não acoplada. Isto traz benefícios em termos de manutenção de SI, uma vez que a separação destes aspectos em componentes faz com que eles possam ser evoluídos separadamente.

As abordagens existentes na literatura, discutidas na Seção 2.2, exigem que, para cada interface a ser construída, sejam modelados e implementados todos os elementos de interação e comportamento requeridos para a interface de usuário. O uso do conceito de Estereótipo de Interface auxilia neste sentido. Uma vez que exista um banco de estereótipos modelados e implementados, a geração de IU torna-se simples, pois apenas a parte que depende dos elementos de negócio devem ser modeladas ou geradas automaticamente, pois o estereótipo já possui a descrição dos demais elementos que são comuns a todas as interfaces com o mesmo propósito.

Quanto ao comportamento, o uso de estereótipos também promove a reutilização, já que as regras de comportamento são implementadas pelo estereótipo. Apenas as regras que devem ser implementadas pela aplicação ou sistemas externos serão diferentes de uma interface para outra.

Estas conclusões foram baseadas na discussão que ilustra a utilização da arquitetura proposta (Capítulo 5), bem como nos Estereótipos de Interface identificados para SI e o emprego da arquitetura baseada em Estereótipos para demonstração da aplicação do trabalho proposto.

Como foi discutido na Seção 2.3, uma abordagem para construção de IU para SI deveria atender alguns requisitos. Na abordagem aqui proposta, o requisitos de Embelezamento é atendido por meio do mecanismo de inserção de CSS no Metamodelo de Apresentação. O requisito de Composição é atendido por meio da modelagem da apresentação dos estereótipos de interface. Na especificação concreta do estereótipo, é desenvolvido um template que contém o leiaute do estereótipo.

A utilização de diferentes modelos, que consideram diferentes aspectos de uma aplicação de SI (IHC, Domínio e Processo de Negócio), proporciona a separação de

Tabela 6.1: *Associação dos Requisitos com a abordagem proposta neste trabalho*

Requisitos	**Forma de tratamento na abordagem proposta**
Embelezamento (Organização)	Inserção de CSS na Modelagem Concreta
Composição (Organização)	Template em JSF
Separação (Organização)	Utilização de diferentes modelos
Propósito (Decomposição)	Diferentes tipos de estereótipos
Agrupamento (Decomposição)	Taxonomia de elementos de IHC
Padronização (Decomposição)	Estereótipo de Interface
Clareza (Mapeamento)	Especificação de Regras de Mapeamento
Flexibilidade (Mapeamento)	Adição de informações em nível concreto
Direção (Mapeamento)	Especificação de Regras de Mapeamento
Generalidade (Abstração)	Metamodelo de IHC
Estrutura (Abstração)	Metamodelos em linguagem UML
Contextualização (Conformidade)	Mapeamento do Modelo de Domínio para Metamodelo de IHC
Correlação (Conformidade)	Integração entre modelos

preocupações, que corresponde ao requisito de Separação. Além disso, a integração entre estes modelos, como apresentada na Seção 5.6, atende ao requisito de Contextualização.

A proposta possibilita, ainda, uma melhor organização da interface por meio do conceito de estereótipo, além de especificar um comportamento padronizado para todas as IU que abstraiam o mesmo propósito. Por meio do conceito de estereótipo podem ser geradas interfaces com diferentes intenções, considerando o requisito Propósito da classe Decomposição.

Como o estereótipo deve ser o elemento dominante da interface, ele ajuda a promover a padronização da aparência e comportamento de IU que empreguem o mesmo estereótipo, atendendo ao requisito de Padronização. Além disso, a taxonomia dos elementos de interação apoia a decomposição dos elementos de estereótipos, podendo agrupar um conjunto de elementos para um objetivo comum. Isto atende ao requisito de Agrupamento.

A especificação das regras de transformação no Apêndice A e do componente para geração automatizada de IU é uma aplicação dos requisitos Clareza e Direção da classe Mapeamento, conforme transformações especificadas para cada passo da construção de IU. A adição de informações no nível concreto sem entrar em conflito com a especificação do estereótipo mostram o requisito de Flexibilidade na geração de IU.

Os metamodelo produzidos, conforme apresentados no Capítulo 3, atendem ao requisito de Estrutura da classe Abstração por serem apresentados em uma linguagem

semi-formal. Ainda com relação a esta classe, a abordagem proposta neste trabalho atende ao requisito de Generalidade, já que traz benefícios por permitir especificar IU independente de uma plataforma por meio do Metamodelo de IHC.

A integração do aspecto de IHC com outros aspectos essenciais para a construção de SI apresentada no Capítulo 5 considera o requisito de Correlação. A utilização do Metamodelo de Domínio e seu mapeamento adequado para uma instância do Metamodelo de IHC atende ao requisito de Contextualização, importante na geração de SI.

6.2 Contribuições

O presente trabalho tem como principal contribuição uma abordagem baseada em modelos para construção automática de IU para SI que permite lidar com os desafios identificados na Seção 1.1. Esta abordagem é composta por três metamodelos [15, 34] integrados em uma arquitetura de um componente de software que automatiza a aplicação da abordagem.

Com base nos desafios apresentados no Capítulo 1, este trabalho investigou abordagens de construção de IU baseada em modelos, contribuindo com uma taxonomia de requisitos para abordagens de construção de IU baseada em modelos para que os desafios citados sejam superados no desenvolvimento de IU para SI.

Outra contribuição importante foi o conjunto de metamodelos proposto, que visa atender às particularidades de cada passo da construção de IU baseada em modelos. O Metamodelo de Domínio é empregado para a Modelagem do Domínio do Negócio de SI, o Metamodelo de IHC para a Modelagem da Interface Abstrata, e o Metamodelo de Apresentação é empregado para a Modelagem da Interface Concreta, possibilitando inserir detalhes de determinada plataforma computacional. Estes metamodelos trabalham em conjunto na arquitetura do componente de Gerência de IU para automatizar a construção de IU.

Uma inovação é a especificação de uma abordagem de construção de IU com o uso de estereótipos [14]. A arquitetura proposta possibilita a geração de IU com base em estereótipos. Além disso, Estereótipos de Interface aplicáveis a SI foram identificados, e uma demonstração da aplicação da abordagem empregando estes estereótipos mostra a viabilidade da abordagem, bem como da utilização da arquitetura no processo de desenvolvimento. A correlação da construção de IU proposta neste trabalho com outros aspectos de um software de SI foram discutidos [19], uma vez que o desenvolvimento de IU não pode ser realizado de maneira isolada.

6.3 Limitações

Este trabalho propõe uma abordagem para construção de IU baseada em modelos com a utilização de uma ferramenta para gerência das IU desenvolvidas. Neste contexto, este trabalho possui algumas limitações. A primeira limitação identificada é que não é possível controlar a forma como o Estereótipo de Interface é modelado. Esta é uma atividade que depende dos projetistas que realizarão esta atividade. Devem ser definidas manualmente a forma de apresentação e a API de regras para conexão da interface com a aplicação ou sistemas externos.

Outra limitação está relacionada com a arquitetura proposta. Esta arquitetura restringe-se a SI convencionais, isto é, IU compostas por janelas, ícones, menos e ponteiros. IU empregados em outros dispositivos podem requerer detalhes específicos que devem ser considerados no momento de geração da IU.

A modelagem de estereótipos traz benefícios de produtividade e consistência para estereótipos que são muito utilizados em SI. Porém modelar um estereótipo que seja utilizado raramente será oneroso para o processo de desenvolvimento de IU.

Além disso, a especificação detalhada das ações é um ponto em que a geração automática ainda não pode atuar de maneira completa. Gerar os métodos em linguagens computacionais ainda é uma tarefa predominantemente realizada de forma manual.

Também deveriam ser acoplados mais padrões de interação aos modelos para a geração automática de IU com maior usabilidade. No entanto, a falta de descrições mais formais de padrões de IHC faz com que sejam necessárias mais pesquisas para que seja avaliado como deve ser eliminada esta lacuna [53].

6.4 Trabalhos Futuros

A pesquisa apresentada neste trabalho traz uma contribuição limitada ao campo de Engenharia de IU baseada em modelos. Outros trabalhos podem estender esta pesquisa, bem como complementá-la.

Este trabalho apresentou o projeto de um componente para gerenciar a construção de IU e partes do componente bem como alguns exemplos de estereótipos foram testados via programação. No entanto, é necessário completar a implementação do componente para Gerência de IU em SI, já que foi implementado o pacote Modelo e realizadas simulações para avaliar a proposta. Este componente também pode ser integrado aos demais componentes do *framework* de gestão de SI. Além disso, experimentos podem ser realizados para verificar a satisfação dos projetistas de SI com o uso do componente.

Apesar de terem sido identificados estereótipos (portal e CRUD), apenas o estereótipo CRUD (composto por formulário e planilha) teve uma versão preliminar

empregado em projetos reais. Avaliar a utilização de estereótipos com a realização de estudos de caso e a utilização do componente em um *framework* para geração de SI em projetos reais de desenvolvimento de software implicam no possível surgimento de novos requisitos não previstos inicialmente, e que poderiam ser implementados através da extensão ou adaptação deste trabalho de pesquisa.

Para auxiliar na manipulação dos metamodelos, um editor visual pode ser desenvolvido, de forma que se considere também os requisitos de personalização e embelezamento de IU.

Esta pesquisa também pode ser ampliada para IU multiplataforma, acrescentando aspectos inerentes a outros dispositivos, que no *framework* CAMELEON são denominados de Contexto de Uso [9].

Questões de usabilidade podem ser acrescentadas aos metamodelos, e estudos podem ser realizados para verificar se as IU geradas com esta abordagem garantem, ou pelo menos auxiliam, a construção de IU com maior qualidade e usabilidade.

Outra extensão relacionada aos estereótipos, é a definição de diretrizes para a identificação e modelagem de estereótipos de interface, de modo a auxiliar o trabalho dos projetistas.

Dentro do ciclo de vida de SI, deve ser possível realizar a evolução de uma interface, permitindo alterações nas informações apresentadas. Porém estas alterações devem respeitar as definições do Estereótipo de Interface. Seria interessante que o componente para Gerência de IU possibilitasse essa evolução das IU.

Como pode-se notar, há ainda muitas questões a serem investigadas na área de Engenharia de IU baseada em modelos para SI. Este trabalho investigou algumas alternativas e abordagens relacionadas a este assunto. Baseado na Engenharia de IU, este trabalho reuniu alguns aspectos importantes em uma única abordagem para gerência de IU, possibilitando a geração de IU não apenas de forma mais eficiente e produtiva, mas com qualidade, reutilização e manutenibilidade durante o ciclo de vida de SI.

Referências Bibliográficas

[1] AHMED, S.; ASHRAF, G. **Model-Based User Interface Engineering with Design Patterns**. *Journal of Systems and Software*, 80:1408 – 1422, August 2007. Elsevier Science Inc.

[2] ALMEIDA, A. C.; BOFF, G.; DE OLIVEIRA, J. L. **A Framework for Modeling, Building and Maintaining Enterprise Information Systems Software**. In: *Anais do XXIII Simpósio Brasileiro de Engenharia de Software*, p. 115 – 125, Fortaleza, Brasil, 2009. IEEE Computer Society.

[3] BASTIEN, J. C.; SCAPIN, D. L. **Ergonomic Criteria for the Evaluation of Human-Computer Interfaces**. Technical report, Institut National de recherche en informatique et en automatique – INRIA Rocquencourt, France, 1993.

[4] BITTAR, T. J.; FORTES, R. P. D. M.; LOBATO, L. L.; WATANABE, W. M. **Web Communication and Interaction Modeling using Model-Driven Development**. In: *Proceeding of ACM International Conference on Design Communication*, p. 193 – 198, Bloomington, Indiana, USA, October 5-7 2009.

[5] BLOUIN, A.; BEAUDOUX, O. **Improving modularity and usability of interactive systems with Malai**. In: *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing System*, p. 115 – 124, Berlin, Germany, June 19-23 2010. ACM.

[6] BOGDAN, C.; FALB, J.; KAINDL, H.; KAVALDJIAN, S.; POPP, R.; HORACEK, H.; ARNAUTOVIC, E.; SZEP, A. **Generating an abstract user interface from a discourse model inspired by human communication**. *Hawaii International Conference on System Sciences*, 0:36 – 45, 2008. IEEE Computer Society.

[7] BORCHERS, J. O. **A pattern approach to interaction design**. In: *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*, DIS '00, p. 369–378, New York City, New York, United States, 2000. ACM.

[8] CADAVID, J. J.; QUINTERO, J. B.; LOPEZ, D. E.; HINCAPIÉ, J. A. **A domain specific language to generate web applications**. In: *Memorias de la XII Conferencia Iberoamericana de Software Engineering (CIbSE 2009)*, p. 139 – 144, Medellín, Colombia, Abril 13-17 2009. CIbSE 2009.

[9] CALVARY, G.; COUTAZ, J.; THEVENIN, D.; LIMBOURG, Q.; BOUILLON, L.; VANDERDONCKT, J. **A Unifying Reference Framework for multi-target user interfaces**. *Interacting with Computers*.

[10] CERI, S.; FRATERNALI, P.; MATERA, M. **Conceptual modeling of data-intensive Web applications**. *IEEE Internet Computing*, 6:20 – 30, July 2002.

[11] COMUNITY, J. **Richfaces**. Available in: http://www.jboss.org/richfaces. Last access: March 1 2011.

[12] COSTA, D.; NÓBREGA, L.; NUNES, N. J. **An MDA Approach for Generating Web Interfaces with UML ConcurTaskTrees and Canonical Abstract Prototypes**. In: *Proc. of 5th International Workshop on Task Models and Diagrams for User Interface Design*, p. 95 – 102, Hasselt, Belgium, 2006. Springer.

[13] COSTA NETO, M.; SOUZA, A.; LAVOR, R.; SILVA, C.; LEITE, J. **Desenvolvendo interfaces de usuário multiplataformas utilizando MDA**. In: *Anais do Primeiro Congresso Regional de Design de Interação*, p. 26 – 34, São Paulo, Novembro 2009. IXDA - SP.

[14] DA COSTA, S. L.; DE OLIVEIRA, J. L. **Construção e Manutenção Baseadas em Modelos de Interfaces para Usuário em Sistemas de Informação**. In: *Anais do III Workshop de Teses e Dissertações de Sistemas de Informação do VI Simpósio Brasileiro de Sistemas de Informação*, Maraba, PA, Brasil, Junho 2010.

[15] DA COSTA, S. L.; GRACIANO NETO, V. V.; LOJA, L. F. B.; DE OLIVEIRA, J. L. **A Metamodel for Automatic Generation of Enterprise Information Systems**. In: *Anais do I Congresso Brasileiro de Software: Teoria e Prática - I Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos*, volume 8, p. 45 – 52, Salvador, BA, Brasil, Setembro 2010. UFBA.

[16] DA SILVA, P. P.; PATON, N. W. **User Interface Modeling in UMLi**. *IEEE Software*, 20:62 – 69, 2003.

[17] DA SILVA, W. C.; DE OLIVEIRA, J. L. **Gerência de Interface Homem-Computador para Sistemas de Informação Empresariais: uma abordagem baseada em modelos**. *iSys - Revista Brasileira de Sistemas de Informação, Vol. 2, 2009*, 2009.

[18] DE OLIVEIRA, J. L.; CUNHA, C. Q.; MAGALHÃES, G. C. **Modelo de Objetos para Construção de Interfaces Visuais Dinâmicas**. In: *Anais do 9o. Simpósio Brasileiro de Engenharia de Software*, p. 143 – 158, Recife, PE, Brasil, 03-06 Outubro 1995. SBC.

[19] DE OLIVEIRA, J. L.; LOJA, L. F. B.; DA COSTA, S. L.; GRACIANO NETO, V. V. **Um componente para gerência de processos de negócio em sistemas de informação**. In: *VII Simpósio Brasileiro de Sistemas de Informação*, Salvador, BA, Brasil, 23 a 25 de Maio 2011. UFBA.

[20] DE OLIVEIRA, K. M. A.; LULA JR, B. **Desenvolvimento evolutivo de interfaces com o usuário em uma abordagem baseada em modelos e múltipla prototipagem: FastInterface**. In: *Sessão de Ferramentas do Simpósio Brasileiro de Engenharia de Software*, Fortaleza, Ceará, Brasil, 05-09 Outubro 2009. SBC.

[21] ENGEL, J. **A model- and pattern-based approach for development of user interfaces of interactive systems**. In: *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing System*, p. 337–340, Berlin, Germany, June 19-23 2010.

[22] FALB, J.; POPP, R.; ROCK, T.; JELINEK, H.; ARNAUTOVIC, E.; KAINDL, H. **Fully-automatic generation of user interfaces for multiple devices from a high-level model based on communicative acts**. In: *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, p. 26 – 35, Washington, DC, USA, 2007. IEEE Computer Society.

[23] FOLMER, E.; WELIE, M. V.; BOSCH, J. **Bridging Patterns: An approach to bridge gaps between SE and HCI**. *Information and Software Technology*, 48:69 – 89, February 2006. Butterworth-Heinemann.

[24] GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns**. Addison-Wesley, Boston, MA, 1995.

[25] GRACIANO NETO, V. V.; DA COSTA, S. L.; DE OLIVEIRA, J. L. **Lições Aprendidas sobre Desenvolvimento Dirigido por Modelos a partir da refatoração de uma ferramenta**. In: *Anais do VIII Encontro Anual de Computação - ENACOMP*, p. 68 – 75, Catalão, Goiás, Brasil, 2010.

[26] ISSA, L.; PÁDUA, C. I. P. S.; RESENDE, R. F.; VIVEIROS, S.; DE ALCÂNTARA DOS SANTOS NETO, P. **Desenvolvimento de interface com usuário dirigida por modelos com geração automática de código**. In: *Memorias de la X Conferencia*

Iberoamericana de Software Engineering (CIbSE 2007), p. 341 – 354, Isla de Marguerita, Venezuela, Mayo 7-11 2007. CIbSE 2007.

[27] JESPERSEN, J. W.; LINVALD, J. **Investigating User Interface Engineering in the Model-Driven Architecture**. In: *Proceedings of the INTERACT 2003 Workshop on Software Engineering and HCI*, p. 63 – 66, Zurich, Switzerland, September 2003. IFIP Press.

[28] KAVALDJIAN, S.; FALB, J.; KAINDL, H. **Generating content presentation according to purpose**. In: *Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics*, SMC'09, p. 2046 – 2051, San Antonio, TX, USA, 2009. IEEE Press.

[29] KRASNER, G.; POPE, S. **A Cookbook for using the Model-View-Controller User Interface Paradigm in Smalltalk-80**. *Journal of Object-Oriented Programming*, 1(3):26 – 49, 1988.

[30] LEE, C.-M. **User Interface Prototype Generation Technique Supporting Usage-Centered Design**. *International Journal of Software Engineering and Knowledge Engineering*, 19(1):23 – 46, 2009. World Scientific Publishing Company.

[31] LEITE, J. C. **A Model-based Approach to Develop Interactive System using IMML**. International Workshop on Task Models and Diagrams for User Interface Design - TAMODIA'06, p. 68 – 81, Berlin, Heidelberg, 2007. Springer-Verlag.

[32] LIMBOURG, Q.; VANDERDONCKT, J. **The Handbook of Task Analysis for Human-Computer Interaction**, chapter Comparing Task Models for User Interface Design. Mahwah, 2003. Lawrence Erlbaum Associates.

[33] LIMBOURG, Q.; VANDERDONCKT, J. **Addressing the Mapping Problem in User Interface Design with UsiXML**. In: *In Proceedings of the 3rd Int. Workshop on Task Models and Diagrams for User Interface Design TAMODIA 2004*, p. 155 – 163. ACM Press, November 15-16 2004. Prague, Czech Republic.

[34] LOJA, L. F. B.; GRACIANO NETO, V. V.; DA COSTA, S. L.; DE OLIVEIRA, J. L. **A Business Process Metamodel for Enterprise Information Systems automatic generation**. In: *Anais do I Congresso Brasileiro de Software: Teoria e Prática - I Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos*, volume 8, p. 37 – 44, Salvador, BA, Brazil, 25 de Setembro 2010. UFBA.

[35] MARTINS, C.; DA SILVA, A. R. **Modeling User Interfaces with the XIS UML Profile**. In: *Proceedings of the 9th International Conference on Enterprise Information Systems*, p. 98 – 104, Funchal, Madeira, Portugal, June 12-16 2007. SciTePress.

[36] MELIÁ, S.; CACHERO, C. **An MDA approach for the development of web applications**. In: *Proc. of 4th International Conference on Web Engineering*, p. 300 – 305, Munich, Germany, June 2004. Springer Link.

[37] MEMMEL, T.; REITERER, H. **Model-Based and Prototyping-Driven User Interface Specification to Support Collaboration and Creativity**. *International Journal of Universal Computer Science (J.UCS). Special Issue on New Trends in Human-Computer Interaction*, 14(19):3217 – 3235, March 2009. J.UCS - Journal of Universal Computer Science.

[38] MOLINA, P. J. **A Review to Model-Based User Interface Development technology**. In: *Proceedings of the First International Workshop on Making model-based user interface design practical: usable and open methods and tools - CADUI*, volume 103, Madeira, Portugal, January 13 2004. CEUR-WS.org.

[39] MORAN, T. P. **The Command Language Grammar: a representation for the user interface of interactive computer systems**. *International Journal of Man-Machine Studies*, 15(1):3 – 50, 1981.

[40] MOREIRA, A. A.; PIMENTA, M. S. **Reuso de interfaces através de padrões concretos de interação**. In: *IHC '06: Proceedings of VII Brazilian symposium on Human factors in computing systems*, p. 41 – 44, Natal, RN, Brazil, 2006. ACM.

[41] MORI, G.; PATERNO, F.; SANTORO, C. **Design and development of multidevice user interfaces through multiple logical descriptions**. *IEEE Transactions on Software Engineering*, 30:507 – 520, August 2004.

[42] MRACK, M.; DE FREITAS MOREIRA, Á.; PIMENTA, M. **Merlin: Interfaces CRUD em tempo de execução**. In: *Sessão de Ferramentas do Simpósio Brasileiro de Engenharia de Software*, p. 79 – 84, Florianópolis, SC, Brasil, 16-20 Outubro 2006.

[43] NAVARRE, D.; PALANQUE, P. A.; LADRY, J.-F.; BARBONI, E. **ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability**. *ACM Transactions on Computer-Human Interaction*, 16(4):18:1–18:5, 2009.

[44] NEIL, T. **Screen Layouts for Rich Internet Applications**. Available in: http://designingwebinterfaces.com/ria-screen-layouts. Last access: February 1 2011.

[45] OBRENOVIC, Z.; STARCEVIC, D. **Model-Driven Development of User Interfaces: Promises and Challenges**. In: *Proc. of the Computer as a Tool at EUROCON 2005*, p. 1259 – 1262, Serbia & Montenegro, Belgrade, November 22-24 2005.

[46] OMG. **MDA Guide Version**. Technical report, June 2003.

[47] ORACLE. **JavaServer Faces Specification**. Available in: http://javaserverfaces-spec-public.java.net/. Last access: March 1 2011.

[48] ORACLE. **Package java.awt**. Available in: http://download.oracle.com/javase/1.4.2/docs/api/java/awt/package-summary.html. Last access: March 1 2011.

[49] ORACLE. **Package javax.swing**. Available in: http://download.oracle.com/javase/1.4.2/docs/api/javax/swing/package-summary.html. Last access: March 1 2011.

[50] PATERNÒ, F. **Towards a UML for Interactive Systems**. In: Little, M.; Nigay, L., editors, *Engineering for Human-Computer Interaction*, volume 2254 de **Lecture Notes in Computer Science**, p. 7 – 18. Springer Berlin / Heidelberg, 2001.

[51] PATERNÒ, F. **Model-based tools for pervasive usability**. *Interacting with Computers*, 17:291 – 315, 2005. Elsevier.

[52] PEDERIVA, I.; VANDERDONCKT, J.; ESPAÑA, S.; PANACH, J. I.; PASTOR, O. **The beautification process in model-driven engineering of user interfaces**. In: *International Conference on Human-Computer Interaction - INTERACT 2007*, p. 411–425, 2007. IFIP Press.

[53] PETRASCH, R. **Model based user interface development with HCI patterns: variatio delectat**. In: *Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems*, PEICS '10, p. 10 – 11, Berlin, Germany, 2010. ACM.

[54] PINHEIRO DA SILVA, P.; GRIFFITHS, T.; PATON, N. W. **Generating User Interface code in a model based user interface development environment**. In: *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, p. 155 – 160, Palermo, Italy, 2000. ACM.

[55] POPP, R.; FALB, J.; ARNAUTOVIC, E.; KAINDL, H.; KAVALDJIAN, S.; ERTL, D.; HORACEK, H.; BOGDAN, C. **Automatic Generation of the Behavior of a User Interface from a High-Level Discourse Model**. *Hawaii International Conference on System Sciences*, 0:1 – 10, 2009. IEEE Computer Society.

[56] PUERTA, A.; EISENSTEIN, J. **Towards a General Computational Framework for Model-Based Interface Development Systems**. In: *Intelligent User Interfaces*, p. 171 – 178. ACM Press, 1999.

[57] RASKIN, J. **The Humane Interface: New Directions for Designing Interactive Systems**. Addison-Wesley, 1st edition, 2000.

[58] SCHLUNGBAUM, E. **Model-based user interface software tools: Current state of declarative models**. Technical report, Graphics, Visualization and Usability Centre, Georgia Institute of Technology, GVU Tech Report, 1996.

[59] SILVA, B.; BARBOSA, S. **Designing Human-Computer Interaction with MoLIC Diagrams – A Practical Guide**. Technical report, PUC-Rio Inf MCC12/07, Computer Science Department, PUC-Rio, Brazil, 2007.

[60] SINNIG, D.; FORBRIG, P.; SEFFAH, A. **Patterns in model-based development**. In: *2nd Workshop on Software and Usability Cross-Pollination - The Role of Usability Patterns, INTERACT 2003*, Zurich, Switzerland, 1st-5th September 2003.

[61] SINNIG, D.; MIZOUNI, R.; KHENDEK, F. **Bridging the gap: Empowering Use Cases with Task Models**. In: *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing System , EICS 2010*, p. 291 – 296, Berlin, Germany, June 19-23 2010.

[62] SMITH, D. M.; PHIFER, G. **Vertical enterprise portals by industry**. Available in: http://www.jboss.org/richfaces. Last access: March 1 2011., January 17 2002.

[63] SOLVBERG, A. **Intentional Perspectives on Information Systems Engineering**, chapter On Roles of Models in Information Systems, p. 17 – 38. Springer Publishing Company, Incorporated, 1st edition, 2010.

[64] SOUCHON, N.; LIMBOURG, Q.; VANDERDONCKT, J. **Task Modelling in Multiple Contexts of Use**. In: *Proceedings of the 9th International Workshop on Interactive Systems. Design, Specification, and Verification*, DSV-IS '02, p. 59 – 73, London, UK, 2002. Springer-Verlag.

[65] SOUSA, K.; MENDONÇA, H.; VANDERDONCKT, J. **A model-driven approach to align business processes with user interfaces**. *International Journal of Universal Computer Science (J.UCS)*, 14(19):3236 – 3249, 2008. J.UCS - Journal of Universal Computer Science.

[66] SZEKELY, P. **Retrospective and Challenges for Model-Based Interface Development**. In: *Design, Specification and Verification of Interactive Systems 96*, p. 1 – 27. Springer-Verlag, 1996.

[67] TEAMDEV. **OpenFaces**. Available in: http://openfaces.org/. Last access: March 1 2011.

[68] TIDWELL, J. **Designing Interfaces: Patterns for Effective Interaction Design**. O'Reilly Media, Incorporated, 2nd edition, 2010.

[69] VALVERDE, F.; PANACH, I.; PASTOR, O. **An abstract interaction model for a MDA software production method**. In: *ER '07: Tutorials, posters, panels and industrial contributions at the 26th international conference on Conceptual modeling*, p. 109–114, Darlinghurst, Australia, 2007. Australian Computer Society, Inc.

[70] VAN WELIE, M. **Interaction Design Pattern Library**. Available in: http://www.welie.com/patterns/index.php. Last access: February 1 2011.

[71] VAN WELIE, M.; VAN DER VEER, G. C. **Pattern languages in interaction design: Structure and organization**. In: *Proceedings of INTERACT 2003*, p. 1 – 5, Zurich, Switzerland, 2003. IOS Press.

[72] VANDERDOCKT, J. **Model-Driven Engineering of User Interfaces: Promises, Successes, Failures, and Challenges**. In: *Proc. of 5th Annual Romanian Conference on Human-Computer Interaction ROCHI'08*, p. 1 – 10, Bucarest, Romania, September 2008. Matrix-ROM.

[73] VANDERDONCKT, J. **A MDA-Compliant Environment for Developing User Interfaces of Information Systems**. In: *Proc. of 17 th Conf. on Advanced Information Systems Engineering CAiSE'05*, p. 16 – 31, Porto, Portugal, June 13-17 2005. Springer.

[74] VANDERDONCKT, J.; PUERTA, A. R. **Introduction to Computer-Aided Design of User Interfaces**. In: *Computer-Aided Design of User Interfaces II, Proceedings of the Third International Conference of Computer-Aided Design of User Interfaces*, p. 1 – 6, Louvain-la-Neuve, Belgium, October 21-23 1999.

[75] W3C. **Cascading Style Sheets Level 2 Revision 1**. Available in: http://www.w3.org/TR/CSS2/cover.html. Last access: March 1 2011.

[76] XUDONG, L.; JIANCHENG, W. **User Interface Design Model**. *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, ACIS International Conference on*, 3:538 – 543, 2007. Los Alamitos, CA, USA.

APÊNDICE A

Regras de Transformação entre Modelos

A.1 Regras de Mapeamento do Metamodelo de Domínio para o Metamodelo de IHC empregando o Estereótipo Formulário

O primeiro passo do mapeamento é associar o Elemento de Negócio ao estereótipo Formulário, conforme mostra a Figura A.1.

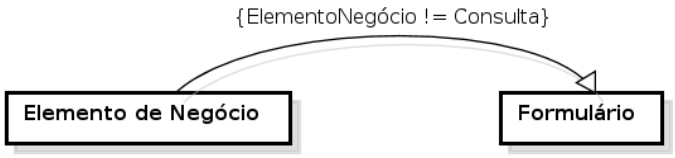


Figura A.1: *Mapeamento do Elemento de Negócio para um Estereótipo Formulário*

Para cada atributo do Elemento de Negócio mapeado para um Formulário, são realizados os mapeamentos representados nas Figuras A.2 a A.7, de acordo com o tipo do atributo.

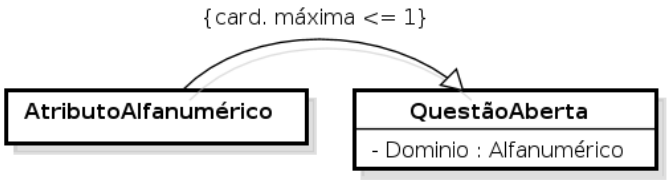


Figura A.2: *Mapeamento de um Atributo Alfanumérico para uma Questão Aberta*

Um atributo alfanumérico é transformado em uma questão aberta, com domínio alfanumérico.

Um atributo numérico é transformado em uma questão aberta, com domínio numérico.

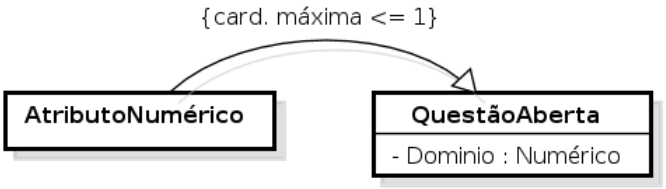


Figura A.3: *Mapeamento de um Atributo Numérico para uma Questão Aberta*

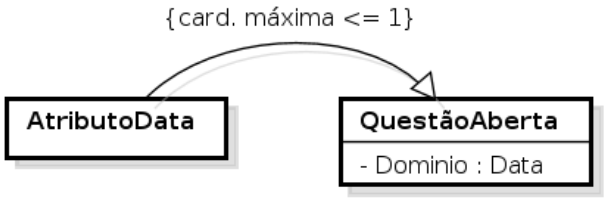


Figura A.4: *Mapeamento de um Atributo Data para uma Questão Aberta*

Um atributo do tipo data é transformado em uma questão aberta, com domínio data.

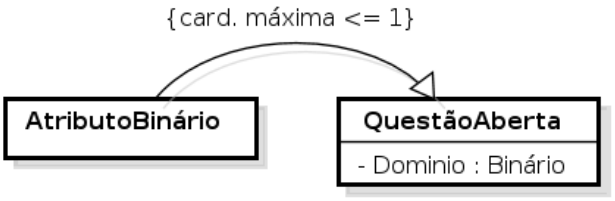


Figura A.5: *Mapeamento de um Atributo Binário para uma Questão Aberta*

Um atributo do tipo binário é transformado em uma questão aberta, com o domínio binário.

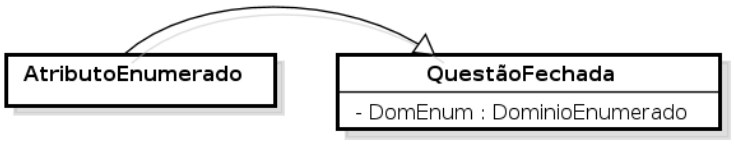


Figura A.6: *Mapeamento de um Atributo Enumerado para uma Questão Fechada*

Um atributo enumerado é transformado em uma questão fechada, com domínio enumerado.

Uma regra é transformada em uma ação externa, com a condição de indicar qual o evento e em qual elemento essa ação deve ser disparada. No metamodelo de IHC, os elementos de interação associados a uma ação externa são do tipo informação de serviço.

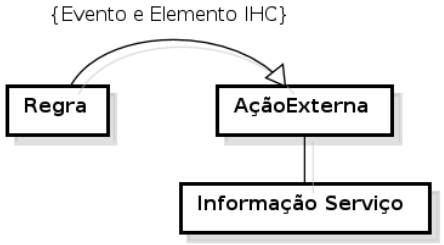


Figura A.7: *Mapeamento de uma Regra para uma Ação Externa*

Logo, o mapeamento das regras do metamodelo de domínio geram, além de uma ação externa, um elemento de informação do tipo serviço.

A.2 Regras de Mapeamento do Metamodelo de IHC para o Metamodelo de Apresentação

As Figuras A.8 a A.21 trazem as possibilidades de mapeamento dos elementos do Metamodelo de IHC para o Metamodelo de Apresentação. Os textos sobre as setas indicam as condições para que ocorra a transformação, mas nem todas as transformações terão condições predefinidas, uma vez que estas condições podem variar de uma plataforma para outra. O Metamodelo de Apresentação foi baseado na Biblioteca OpenFaces. Os paineis do Metamodelo de Apresentação aqui apresentados correspondem aos *ManagedBeans* da Figura 4.7.

Figura A.8: *Mapeamento do Estereótipo de Interface para Template*

Para mapear um Estereótipo para um Template (Figura A.8), devem ser inseridas informações em nível concreto, como leiaute e estilo.

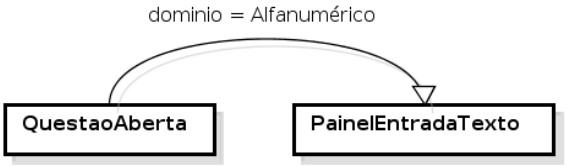


Figura A.9: *Mapeamento da QuestaoAberta para PainelEntradaTexto*

Uma questão aberta é mapeada para um painel de entrada do tipo texto se o domínio dela for alfanumérico. Da mesma forma, uma questão aberta com domínio data é mapeada para um painel de entrada do tipo data.

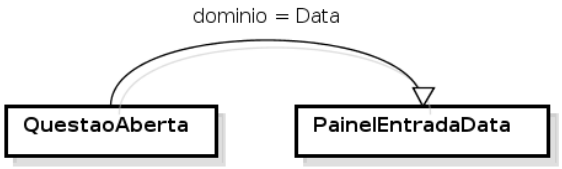


Figura A.10: *Mapeamento da QuestaoAberta para PainelEntradaData*

Uma questão fechada é mapeada para painéis de entrada do tipo enum. As questões fechadas são mapeadas para painéis de entrada do tipo boolean se a questão tiver cardinalidade máxima igual a um e os valores do domínio enumerado forem no máximo 2 (verdadeiro e falso).

Se houver mais de um valor para o domínio enumerado então deve ser mapeado para o painel entrada radio. Se a cardinalidade máxima de uma questão fechada for maior que um, esta é mapeada para um painel entrada do tipo lista por padrão, ou pode ser mapeada para o tipo manycheck (a diferença está na forma de apresentação), conforme a escolha do projetista.

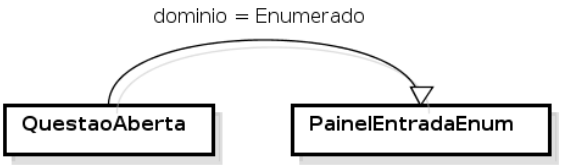


Figura A.11: *Mapeamento da QuestaoFechada para PainelEntradaEnum*

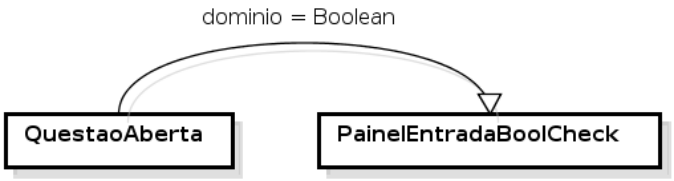


Figura A.12: *Mapeamento da QuestaoFechada para PainelEntradaBoolean*

Para os elementos do tipo informação, um serviço é mapeado para um painel navegação do tipo botão, e este deve ser associado ao acionamento de uma regra. No metamodelo abstrato, o serviço está associado a uma ação, e neste mapeamento esta ação é associada como uma regra ao painel.

Se a regra for de interface, o serviço pode também ser mapeado para um painel navegação do tipo link ou do tipo menu, conforme as decisões do projetista.

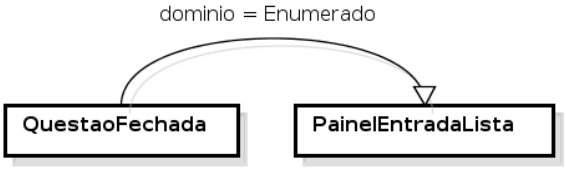


Figura A.13: *Mapeamento da QuestaoFechada para PainelEntradaLista*

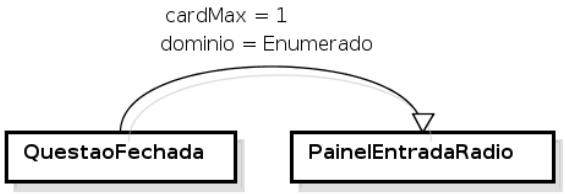


Figura A.14: *Mapeamento da QuestaoFechada para PainelEntradaRadio*

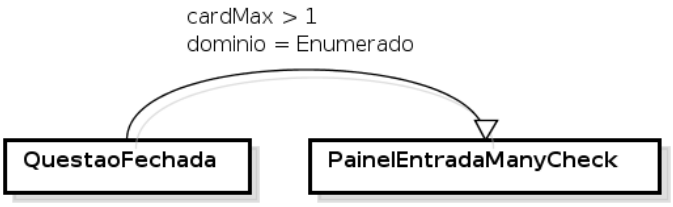


Figura A.15: *Mapeamento da QuestaoFechada para PainelEntradaChecarMultiplos*

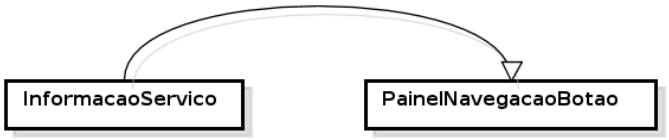


Figura A.16: *Mapeamento de InformacaoServico para PainelNavegacaoBotao*

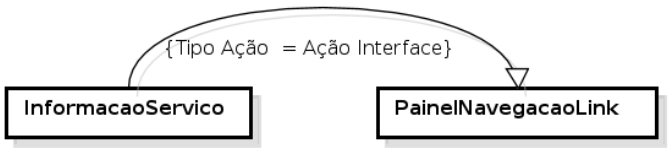


Figura A.17: *Mapeamento de InformacaoServico para PainelNavegacaoLink*

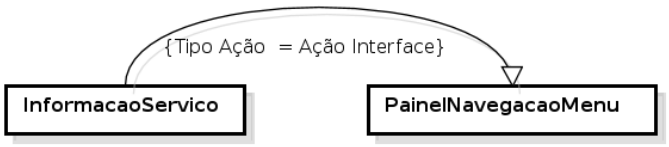


Figura A.18: *Mapeamento de InformacaoServico para PainelNavegacaoMenu*

Elementos de informação do tipo *feedback* são mapeados para painel informação *feedback*. Informação textual mapeia-se para painel de informação textual.

População é mapeada para painel informação população, devendo ser ainda passado qual elemento de negócio conterá a população a ser exibida.

Figura A.19: *Mapeamento de InformacaoFeedback para PainelInfoFeedback*

Figura A.20: *Mapeamento de InformacaoTextual para PainelInfoTextual*

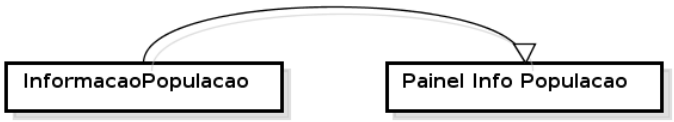


Figura A.21: *Mapeamento de InformacaoPopulacao para PainelInfoPopulacao*