

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

AUGUSTO ABELIN MOREIRA

**Reuso de IHC orientado a padrões concretos
de interação e dirigido por casos de uso,
~~dirigido por casos de uso e integrado a um~~
~~processo de desenvolvimento baseado em~~
UML**

Dissertação apresentada como requisito parcial
para a obtenção do grau de Mestre em Ciência
da Computação

Prof. Dr. Marcelo Soares Pimenta
Orientador

Porto Alegre, novembro de 2006.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Moreira, Augusto Abelin

Reuso de IHC orientado a padrões concretos de interação e dirigido por casos de uso / Augusto Abelin Moreira – Porto Alegre: Programa de Pós-Graduação em Computação, 2006.

15 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2006. Orientador: Marcelo Soares Pimenta.

1.Reuso. 2.IHC. 3.Caso de Uso. I. Moreira, Augusto Abelin. II.Pimenta, Marcelo Soares. III. Reuso de IHC orientado a padrões concretos de interação e dirigido por casos de uso.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. José Carlos Ferraz Hennemann

Vice-reitor: Prof. Pedro Cezar Dutra Fonseca

Pró-Reitora de Pós-Graduação: Profa. Valquiria Linck Bassani

Diretor do Instituto de Informática: Prof. Flávio Rech Wagner

Coordenador do PPGC: Prof. Carlos Alberto Heuser

Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Agradeço às mulheres da minha vida - minha esposa Karin e minha filha Ana Luiza
- pelo apoio, incentivo e compreensão ao longo desta jornada.

SUMÁRIO

ABSTRACT.....	14
1 INTRODUÇÃO.....	15
2 REUSO DE SOFTWARE: DEFINIÇÕES E PRÁTICAS.....	17
2.1 Reuso: conceito.....	17
2.2 Benefícios do reuso.....	17
2.3 Problemas do reuso.....	18
2.4 Práticas de reuso na engenharia de software.....	19
2.5 Assets: Uma especificação de artefatos reusáveis.....	22
2.6 Processos de reuso.....	26
3 DESENVOLVIMENTO DE SOFTWARE COM UML E CASOS DE USO.....	30
3.1 O uso da UML para a modelagem de sistemas.....	30
3.2 Casos de uso: fundamentos.....	31
3.2.1 Conceito.....	31
3.2.2 Elementos.....	33
3.2.3 Estrutura básica de descrição.....	41
3.3 Processo de desenvolvimento dirigido a casos de uso	41
3.4 Abordagens para reuso de casos de uso.....	43
3.4.1 Casos de uso reusáveis.....	43
3.4.2 Casos de uso parametrizados.....	45
3.4.3 Padrões de casos de uso (use case patterns).....	45
3.4.4 Padrões de domínio.....	48
4 PROJETO E REUSO DE INTERFACES COM O USUÁRIO.....	49
4.1 Projeto de interface com o usuário no processo de desenvolvimento.....	49
4.2 Modelos de especificação de interface com o usuário.....	51
4.2.1 Protótipos.....	51
4.2.2 Modelos de Interface de Usuário.....	55
4.2.3 Um modelo de referência para IUs.....	57
4.2.4 Linguagens de descrição de interfaces com o usuário	61
4.3 Abordagens para reuso de IHC.....	65
4.3.1 Recomendações e guias de estilo.....	65
4.3.2 Padrões de interação e linguagens de padrões.....	66
4.3.3 Geradores de interface.....	72
4.3.4 Componentização.....	73
5 ABORDAGEM PROPOSTA.....	74

5.1 Árvore de Reificação de IU.....	75
5.2 Padrão Concreto de Interação (CIP).....	79
5.3 Padrão Concreto de Caso de Uso (CUCP).....	81
5.4 Padrão de Interface de Gerenciamento do Sistema (SMUIP).....	84
5.5 Processos de reuso na abordagem proposta.....	87
5.5.1 Processo de consumo.....	88
5.5.2 Processo de identificação.....	93
5.5.3 Processo produção.....	95
5.5.4 Processo de gerenciamento.....	96
5.6 Considerações sobre ferramentas de suporte à abordagem.....	97
5.6.1 Modelagem de casos de uso e demais modelos da UML.....	97
5.6.2 Modelagem de IU.....	98
5.6.3 Repositório de assets	98
5.6.4 Ferramentas de transformação	98
6 EXEMPLO DE APLICAÇÃO.....	99
6.1 Planejamento do estudo de caso.....	99
6.1.1 Objetivo.....	99
6.1.2 Características da empresa onde foi executado o estudo de caso	99
6.1.3 Escopo da aplicação da abordagem.....	101
6.1.4 Plano de atividades.....	101
6.2 Descrição do sistema a ser desenvolvido.....	102
6.3 Identificação dos artefatos reusáveis.....	102
6.4 Praticando o reuso.....	103
6.4.1 Análise de reuso de casos de uso.....	103
6.4.2 Análise de reuso de interação.....	106
6.5 Considerações sobre os resultados obtidos.....	112
6.5.1 Quanto às limitações do estudo de caso.....	113
6.5.2 Quanto à integração dos processos de reuso no processo de desenvolvimento...	113
6.5.3 Quanto à produtividade no processo de desenvolvimento.....	113
6.5.4 Quanto aos efeitos na qualidade da interação.....	120
7 CONCLUSÃO.....	121
7.1 Contribuições.....	122
7.2 Perspectivas de trabalhos futuros.....	123
7.3 Lista de publicações.....	124

LISTA DE ABREVIATURAS E SIGLAS

<u>ABD</u>	<u>Asset-Based Development</u>
<u>AIO</u>	<u>Abstract Interaction Object</u>
<u>AUI</u>	<u>Abstract User Interface</u>
<u>CASE</u>	<u>Computer Aided Software Engineering</u>
<u>CIO</u>	<u>Concrete Interaction Object</u>
<u>CIP</u>	<u>Concrete Interaction Pattern</u>
<u>CMS</u>	<u>Content Manegement System</u>
<u>COTS</u>	<u>Commercial-Off-The-Shelf software</u>
<u>CRF</u>	<u>Cameleon Reference Framework</u>
<u>CRUD</u>	<u>Create, Read, Update and Delete</u>
<u>CUCP</u>	<u>Concrete Use Case Pattern</u>
<u>CUI</u>	<u>Concrete User Interface</u>
<u>DTD</u>	<u>Document Type Definition</u>
<u>EP</u>	<u>Effort Proportion</u>
<u>ES</u>	<u>Engenharia de Software</u>
<u>EUCP</u>	<u>Extended Use Case Point</u>
<u>FUI</u>	<u>Final User Interface</u>
<u>GUI</u>	<u>Graphical User Interface</u>
<u>IDE</u>	<u>Integrated Development Environment</u>
<u>IHC</u>	<u>Interação Humano-Computador</u>
<u>IO</u>	<u>Interaction Object</u>
<u>IP</u>	<u>Interaction Pattern</u>
<u>IU</u>	<u>Interface do Usuário</u>
<u>MDA</u>	<u>Model Driven Architecture</u>
<u>OMG</u>	<u>Object Management Group</u>
<u>OO</u>	<u>Orientação a Objetos</u>
<u>PIM</u>	<u>Platform Independent Model</u>

<u>PLML</u>	<i><u>Pattern Language Markup Language</u></i>
<u>POA</u>	<i><u>Programação Orientada a Aspectos</u></i>
<u>PSM</u>	<i><u>Platform Specific Model</u></i>
<u>PU</u>	<u>Processo Unificado</u>
<u>RAS</u>	<i><u>Reusable Asset Specification</u></i>
<u>RE</u>	<i><u>Reuse Factor</u></i>
<u>SMUI</u>	<i><u>System Management User Interface</u></i>
<u>SMUIP</u>	<i><u>System Management User Interface Pattern</u></i>
<u>SsD</u>	<u>Sistema sendo Demandado</u>
<u>SUI</u>	<i><u>Sketched User Interface</u></i>
<u>T&C</u>	<i><u>Task & Concepts</u></i>
<u>TI</u>	<u>Tecnologia da Informação</u>
<u>UC</u>	<i><u>Use Case</u></i>
<u>UI</u>	<i><u>User Interface</u></i>
<u>UIDL</u>	<i><u>User Interface Description Language</u></i>
<u>UML</u>	<i><u>Unified Modeling Language</u></i>
<u>XUI</u>	<i><u>eXecutable User Interface</u></i>
<u>ABD</u>	<i><u>Asset-Based Development</u></i>
<u>AIO</u>	<i><u>Abstract Interaction Object</u></i>
<u>AUI</u>	<i><u>Abstract User Interface</u></i>
<u>CASE</u>	<i><u>Computer-Aided Software Engineering</u></i>
<u>CIO</u>	<i><u>Concrete Interaction Object</u></i>
<u>CIP</u>	<i><u>Concrete Interaction Pattern</u></i>
<u>CMS</u>	<i><u>Content Manegement System</u></i>
<u>CRF</u>	<i><u>Cameleon Reference Framework</u></i>
<u>CUI</u>	<i><u>Concrete User Interface</u></i>
<u>EP</u>	<i><u>Effort Proportion</u></i>
<u>ES</u>	<u>Engenharia de Software</u>
<u>EUCP</u>	<i><u>Extended Use Case Point</u></i>
<u>FUI</u>	<i><u>Final User Interface</u></i>
<u>GUI</u>	<i><u>Graphical User Interface</u></i>
<u>IDE</u>	<i><u>Integrated Development Environment</u></i>
<u>IHC</u>	<u>Interação Humano-Computador</u>
<u>IO</u>	<i><u>Interaction Object</u></i>
<u>IP</u>	<i><u>Interaction Pattern</u></i>

IU	Interface do Usuário
OMG	<i>Object Management Group</i>
OO	Orientação a Objetos
PU	Processo Unificado
RAS	<i>Reusable Asset Specification</i>
RF	<i>Reuse Factor</i>
RUCP	<i>Reified Use Case Pattern</i>
SMUI	<i>System Management User Interface</i>
SMUIP	<i>System Management User Interface Pattern</i>
SsD	Sistema sendo Demandado
SUI	<i>Sketched User Interface</i>
T&C	<i>Task & Concepts</i>
TI	Tecnologia da Informação
UI	<i>User Interface</i>
UIDL	<i>User Interface Description Language</i>
UML	<i>Unified Modeling Language</i>
XUI	<i>eXecutable User Interface</i>

LISTA DE FIGURAS

FIGURA 2.1: CATEGORIAS DE REUSO.....	22
FIGURA 2.2: DIMENSÕES DE ASSETS REUSÁVEIS.....	23
FIGURA 2.3: PRINCIPAIS SEÇÕES DO CORE RAS.....	25
FIGURA 2.4: OS FLUXOS DE TRABALHO DO CICLO DE VIDA DE UM ARTEFATO REUSÁVEL.....	26
FIGURA 2.5: DIAGRAMA DE ESTADOS DO CICLO DE VIDA DE UM ASSET.	28
FIGURA 3.1: EXEMPLOS DE ATORES NA UML.....	33
FIGURA 3.2: EXEMPLOS DE CASOS DE USO NA UML.....	34
FIGURA 3.3: FLUXOS DE UM CASO DE USO.....	35
FIGURA 3.4: NOTAÇÃO DA UML PARA RELACIONAMENTO DE ASSOCIAÇÃO.....	35
FIGURA 3.5: CASO DE USO BASE QUE INCLUI COMPORTAMENTO DE OUTRO CASO DE USO.....	36
FIGURA 3.6: EXEMPLO DE RELACIONAMENTO DE INCLUSÃO.....	37
FIGURA 3.7: CASO DE USO BASE COM UM CASO DE USO ESTENDIDO.....	37
FIGURA 3.8: EXEMPLO DE RELACIONAMENTO DE EXTENSÃO.....	38
FIGURA 3.9: NOTAÇÃO DA UML PARA RELACIONAMENTO DE GENERALIZAÇÃO ENTRE CASOS DE USO.....	38
FIGURA 3.10: NOTAÇÃO DA UML PARA RELACIONAMENTO DE GENERALIZAÇÃO ENTRE ATORES.....	39
FIGURA 3.11: DIAGRAMA DE CASOS DE USO.....	40
FIGURA 3.12: METAMODELO DE CASO DE USO.....	40

FIGURA 3.13: O MODELO DE DESENVOLVIMENTO ITERATIVO E INCREMENTAL.....	41
FIGURA 3.14: O MODELO DE CASOS DE USO E SEUS RELACIONAMENTOS COM OUTROS MODELOS DE DESENVOLVIMENTO.	42
FIGURA 3.15: O MODELO DE REQUISITOS “EIXO-E-RAIOS”	43
FIGURA 3.16: EXEMPLO DE PADRÃO DE CASO DE USO.....	48
FIGURA 4.1: EXEMPLO DE PROTÓTIPO EM PAPEL.....	53
FIGURA 4.2: EXEMPLO DE PROTÓTIPO DESENHADO POR COMPUTADOR.....	53
FIGURA 4.3: OBJETOS DE INTERAÇÃO ABSTRATOS E CONCRETOS.....	56
FIGURA 4.4: TIPOS DE OBJETOS DE INTERAÇÃO.....	57
FIGURA 4.5: O CAMELEON REFERENCE FRAMEWORK.....	58
FIGURA 4.6: EXEMPLO DE TRANSFORMAÇÕES NO CRF.....	60
FIGURA 4.7: O MODELO MIM.....	64
FIGURA 5.1: O CAMELEON REFERENCE FRAMEWORK ESTENDIDO.....	76
FIGURA 5.2: EXEMPLO DE SUI	77
FIGURA 5.3: EXTRATO DE UMA CUI PARA O PADRÃO “PARTS SELECTOR”	78
FIGURA 5.4: DIAGRAMA DE CLASSES DO MODELO DE IU.....	79
FIGURA 5.5: ESTRUTURA DE UM PADRÃO CONCRETO DE INTERAÇÃO.	81
FIGURA 5.6: EXEMPLO DE ÁRVORE DE REIFICAÇÃO DE UM CIP.....	81
FIGURA 5.7: METAMODELO DE CASO DE USO ESTENDIDO.....	82
FIGURA 5.8: MAPEAMENTO ENTRE CASOS DE USO E INTERFACES COM O USUÁRIO.....	83
FIGURA 5.9: ESTRUTURA DE UM PADRÃO CONCRETO DE CASO DE USO.....	84
FIGURA 5.10: MAPEAMENTO ENTRE PADRÕES DE CASOS DE USO E PADRÕES DE INTERAÇÃO.....	84
FIGURA 5.11: SYSTEM MANAGEMENT USER INTERFACE - SMUI.....	85

FIGURA 5.12: EXEMPLO DE SMUI WEB.....	86
FIGURA 5.13: PROCESSO DE ANÁLISE DE REUSO INTEGRADO AO PROCESSO DE DESENVOLVIMENTO.....	89
FIGURA 5.14: ARTEFATOS DE IU REUSÁVEIS COMO ASSETS RAS.....	95
FIGURA 6.1: PROTÓTIPO DE IU DO CASO DE USO “PUBLICA NOTÍCIA”.	108
FIGURA 6.2: PROTÓTIPO DE IU DO CASO DE USO “INCLUI NOTÍCIA” - PARTE 1.....	110
FIGURA 6.3: PROTÓTIPO DE IU DO CASO DE USO “INCLUI NOTÍCIA” - PARTE 2.....	111
FIGURA 6.4: PROTÓTIPO DE IU DO CASO DE USO “INCLUI NOTÍCIA” - PARTE 3.....	111
FIGURA 6.5: FÓRMULA PARA OBTER OS PONTOS POR CASO DE USO	114
FIGURA 6.6: FÓRMULA PARA OBTER OS PONTOS POR CASO DE USO EM PROJETOS COM REUSO.....	115
FIGURA 6.7: FÓRMULA DE CÁLCULO DE PONTOS POR CASO DE USO POR FASE DO PROCESSO DE DESENVOLVIMENTO.....	116
FIGURA 6.8: FÓRMULA DE CÁLCULO DE FATOR DE REUSO (RF).....	118

LISTA DE TABELAS

TABELA 2.1. ESTRUTURA PARA PESQUISA DE REUSO DE SOFTWARE	18
TABELA 2.2: CATEGORIAS DE REUSO.....	19
TABELA 3.1: EXEMPLO DE UM CASO DE USO ESSENCIAL.....	32
TABELA 3.2: EXEMPLO DE UM CASO DE USO REAL.....	32
TABELA 3.3: EXEMPLO DE CASO DE USO PARAMETRIZADO.....	45
TABELA 3.4: EXEMPLO DE CASO DE USO APÓS SUBSTITUIÇÃO DE PARÂMETROS.....	45
TABELA 4.1: EFICÁCIA RELATIVA DE PROTÓTIPOS DE BAIXA E DE ALTA-FIDELIDADE.....	55
TABELA 4.2: UM EXEMPLO DE PADRÃO DE INTERAÇÃO.....	68
TABELA 4.3: UM SURVEY DE COLEÇÕES DE PADRÕES DE IHC.....	71
TABELA 5.1: ARTEFATOS REUSÁVEIS EM CADA FASE DO PROCESSO DE DESENVOLVIMENTO.....	88
TABELA 5.2: NÍVEIS DE REUSO DE IHC DE UM CASO DE USO.....	90
TABELA 5.3: EXEMPLO DE LISTA ATOR-OBJETIVO DE UM SISTEMA DE COMÉRCIO ELETRÔNICO.....	91
TABELA 6.1: RESUMO DO INVENTÁRIO DE ASSETS REUSÁVEIS.....	103
TABELA 6.2: LISTA ATOR-OBJETIVO DO SISTEMA EM ESTUDO.....	104
TABELA 6.3: NARRATIVA DO CASO DE USO 52 - PUBLICA NOTÍCIA....	107
TABELA 6.4: CIPS UTILIZADOS NA IU DO CASO DE USO 52 - PUBLICA NOTÍCIA.....	108
TABELA 6.5: NARRATIVA DO CASO DE USO 49 – INCLUI NOTÍCIA.....	109

TABELA 6.6: CIPS UTILIZADOS NA IU DO CASO DE USO 49 – INCLUI NOTÍCIA.....	112
TABELA 6.7: PROPORÇÃO DE ESFORÇO DE DESENVOLVIMENTO NA PROCERGS.....	116
TABELA 6.8: TABELA DE FATOR DE REUSO.....	117
TABELA 6.9: EXEMPLO DE CÁLCULO DE RF E EUCP.....	118
TABELA B.1: PADRÕES DE DOMÍNIO DO ESTUDO DE CASO.....	138
TABELA C.1: PADRÕES CONCRETOS DE CASOS DE USO DO ESTUDO DE CASO.....	139
TABELA D.1: PADRÕES CONCRETOS DE INTERAÇÃO DO ESTUDO DE CASO.....	148
TABELA F.1: EUCP DOS CASOS DE USO DO ESTUDO DE CASO.....	174
TABELA G.1: EXEMPLO DE CASO DE USO APRESENTADO.....	176

RESUMO

A prática de reuso, apesar de ser universalmente e reconhecidamente aceita como uma boa prática, ainda não atingiu um estágio de maturidade satisfatório nas instituições. Em particular, a definição e a construção das interfaces com o usuário (IHC) ainda carecem de abordagens efetivas e sistemáticas de reuso, mesmo se comparadas com as práticas já existentes de reuso dos demais artefatos de software.

O objetivo deste trabalho é duplo: investigar abordagens de reuso de interfaces do usuário (IHC), analisando-as do ponto de vista da sua aplicabilidade, adequação e integração em um processo de desenvolvimento de software baseado na UML e propor uma abordagem de reuso de artefatos de software iterativo dirigida por casos de uso, integrando – por meio de alguns aspectos do ciclo de vida de casos de uso (da modelagem à implementação) – vários conceitos e técnicas de reuso bem conhecidos como padrões de casos de uso, padrões de interação e padrões de projeto. O foco principal é a definição de como promover o reuso de interface com o usuário integrado ao reuso dos artefatos orientados ao domínio da aplicação e de como implementar alguns destes padrões através do reuso de código e/ou de componentes.

Palavras-Chave: reuso de software, reuso de interface com o usuário, reuso dirigido por casos de uso, padrão concreto de interação, padrão concreto de caso de uso ~~de caso de uso reificado~~, caso de uso, padrão de interação, linguagem de padrões, padrão de caso de uso, apresentação de caso de uso, engenharia de software, IHC, UML.

A concrete interaction pattern oriented and use case driven HCI reuse approach
~~pattern-oriented and use case driven HCI reuse approach in-~~
~~tegrated with a development process based on UML~~

ABSTRACT

Reuse practicing, apart from being universally and recognizably accepted as good practice, hasn't reached a fair maturity level in organizations. Specifically, the definition and construction of user interfaces (HCI) lack from effective and systematic reuse approaches even when compared with other reuse practices used nowadays.

This work has a dual-purpose: making a survey of reuse approaches of user interfaces (HCI) and their applicability in a development process based on UML, and to present an use case driven software reuse approach for interactive systems, integrating – by means of some aspects of use case life cycle (from modeling to implementation) - several well-known reuse concepts and techniques like use case patterns, interaction patterns and design patterns. The approach focuses on how to promote user interface reuse integrated to reuse of application-domain related software artifacts and how to implement some of these patterns through code reuse and/or component reuse.

Keywords: software reuse, user interface reuse, use case driven reuse, concrete interaction pattern, ~~concrete~~~~reified~~ use case pattern, use case, interaction pattern, pattern language, use case pattern, use case presentation, software engineering, HCI, UML.

1 INTRODUÇÃO

Desde os primórdios da Engenharia de Software [McIlroy 1968] o reuso de software é visto por muitos autores como a mais provável “bala de prata” para solucionar os problemas do desenvolvimento de sistemas [Cox 1992].

De fato, o reuso de software é um tema da engenharia de software que tem recebido destaque especial nas últimas décadas e tem sido o ‘Santo Graal’ que a orientação a objetos (abreviada OO) vem utilizando para aumentar o número de seus iniciados. No entanto, fatores técnicos, organizacionais, psicológicos, sociológicos e econômicos se interpõem à adoção do reuso em toda a sua capacidade [Tracz 1988]. Para que se façam disponíveis os artefatos que serão reutilizados, antes é necessário que haja uma cultura de reuso interna na organização [Frakes & Isoda 1994], ou um ambiente favorável à produção desses artefatos [Cox 1990] pois reuso de software implica muitas vezes um trabalho extra. Construir artefatos reusáveis de fato requer atividades de identificação, extração, organização e representação destes artefatos de uma maneira que seja fácil de entendê-los e manipulá-los. Encontrar os artefatos reusáveis que possam ser (re)utilizados para o desenvolvimento de um novo sistema pode ser mais difícil e trabalhoso do que o desenvolvimento (do zero) deste novo sistema. De fato, software para ser reusável e reusado deve ser projetado, documentado e implementado para este fim informações de identificação, extração, organização e representação de uma maneira que seja fácil de entender e manipular. Encontrar os artefatos reusáveis que possam ser (re)utilizados para o desenvolvimento de um novo sistema pode ser mais difícil e trabalhoso do que o desenvolvimento (do zero) deste novo sistema. De fato, software para ser reusável e reusado deve ser projetado, documentado e implementado para este fim de reuso [Tracz 1988].

Embora a tecnologia e a notação disponíveis habilitem os desenvolvedores de software a praticar reuso, isto não implica que o reuso ocorrerá. É preciso – além da linguagem padronizada para comunicação (que já existe: UML) – uma estratégia de reuso (com conjunto de conceitos e ferramentas associados) que propicie o reuso em diferentes níveis de abstração.

Tal estratégia deve focar em reuso durante todo o ciclo de desenvolvimento, prover suporte à reutilização de artefatos (*development with reuse*) e à produção de artefatos reusáveis (*development for reuse*), ser facilmente integrável a outros métodos e técnicas de desenvolvimento utilizados e ser implementada por meio de ferramentas (também integráveis a ferramentas correntemente utilizadas) [Sindre et al. 1995].

Uma vez que UML é uma linguagem padronizada pelo OMG, adotada na indústria (e na academia), largamente aplicável e suportada por ferramentas, ela pode ser usada como a fundação desta estratégia. E, como as principais abordagens de processo de desenvolvimento OO promovem e incentivam que o processo seja conduzido por casos de uso, uma estratégia de reuso baseada em casos de uso torna-se promissora.

O objetivo deste trabalho é propor uma abordagem de reuso de artefatos de software - com foco nos artefatos relativos a Interface com Usuário - dirigida por casos de uso. Possibilidades de reuso são analisadas sempre sob a perspectiva do caso de uso ao longo de todo o seu ciclo de vida.

O trabalho está estruturado da seguinte forma: no capítulo 2, fazemos uma discussão sobre a prática de reuso em sistemas interativos. O capítulo 3 faz uma revisão conceitual de casos de uso, do seu papel em um processo de desenvolvimento dirigido por casos de uso, e das práticas de reuso envolvendo casos de uso. A seguir, no capítulo 4, inspecionamos as atividades de projeto de interfaces com o usuário em um processo de desenvolvimento OO, bem como uma revisão das principais propostas de reuso de IHC existentes. Os modelos de artefatos de reuso de IHC propostos pela nossa abordagem e o processo de reuso dirigido por caso de uso são apresentados no capítulo 5. Um exemplo em um contexto real de uso é apresentado no capítulo 6, para ilustrar como se utiliza a abordagem proposta, seguido por uma revisão no capítulo 3 das atividades de projeto de interfaces em um processo de desenvolvimento OO. O capítulo 4 apresenta os conceitos fundamentais para a aplicação da nossa abordagem juntamente com uma revisão das principais propostas de reuso de IHC existentes. O processo de reuso dirigido por caso de uso e os modelos de IU propostos pela nossa abordagem são apresentados no capítulo 5. Um exemplo de aplicação real em um contexto real de uso é apresentado no capítulo 6, para ilustrar como se utiliza a abordagem proposta e para mostrar exemplos de sua aplicabilidade. Finalmente, o capítulo 7 conclui apresentando alguns pontos a serem investigados.

2 REUSO DE SOFTWARE: DEFINIÇÕES E PRÁTICAS

Neste capítulo, são apresentados os conceitos de reuso de software e uma descrição sucinta das principais abordagens de reuso utilizadas no processo de desenvolvimento de software. Também é feita uma revisão dos principais processos relacionados a artefatos reusáveis, os quais sejam: identificação, produção, consumo e gerenciamento, bem como é apresentada uma proposta, baseada em padrões abertos, para a documentação e armazenamento destes artefatos.

2.1 Reuso: conceito

No processo de desenvolvimento de um software, é comum acontecer que problemas similares ~~sejam~~ resolvidos diversas vezes por desenvolvedores diferentes. O princípio básico de reuso de software é não “reinventar a roda”, ou seja, não resolver problemas similares várias vezes.

O primeiro passo para se ter sucesso na prática de reuso é entender que existe mais de uma opção à disposição, pois se pode reusar elementos de diferentes níveis de abstração tais como códigos-fonte, componentes, artefatos de desenvolvimento, padrões e templates, entre outros [Ambler 2006]~~mais de uma opção à disposição, pois se pode reusar elementos de diferentes níveis de abstração tais como códigos-fonte, componentes, artefatos de desenvolvimento, padrões e templates, entre outros [Ambler 2004].~~

O reuso de software não torna a atividade de desenvolvimento mais fácil, muito pelo contrário, vários autores afirmam que o reuso de software demanda um trabalho extra.

Construir artefatos reusáveis requer informações de identificação, extração, organização e representação de uma maneira que seja fácil de entendê-los e manipulá-los. Encontrar os artefatos reusáveis que possam ser utilizados para o desenvolvimento de um novo sistema pode ser mais difícil e trabalhoso do que o desenvolvimento deste novo sistema, pois um software, para ser reusável e reusado deve ser projetado, documentado e implementado visando o reuso [Tracz 1988]. Além disso, ele deve ser escrito e armazenado de uma maneira que permita a fácil compreensão, indexação e busca.

2.2 Benefícios do reuso

A maior parte dos artigos sobre reuso de software não ~~apresentam-se dá ao trabalho de realizar~~ uma lista exaustiva dos benefícios do reuso, possivelmente porque a noção de reuso já é tão arraigada que muitos autores consideram essa explicação dispensável. No entanto, alguns benefícios trazidos pelo reuso não são intuitivos.

O benefício direto do reuso de componentes de software, consequência de sua própria definição, é permitir aos desenvolvedores usar uma quantidade menor de ~~artefatos no desenvolvimento de um sistema e gastar menos tempo organizando esses artefatos~~~~símbolos no desenvolvimento de um sistema e gastar menos tempo organizando esses símbolos~~. Isso tem o efeito direto de aumentar a produtividade, reduzir custos, e melhorar a chance de atender ao cronograma do projeto.

Além disso, como os recursos reaproveitáveis do software devem ser ~~exaustivamente testados e verificados, o reuso tem também o potencial de melhorar a qualidade, a confiabilidade, rigorosamente testados e verificados, o reuso tem também o potencial de melhorar a qualidade, a confiabilidade,~~ a manutenibilidade e a portabilidade do software [Yongbeom & Stohr 1998].

Da mesma forma, em [Meyer 1997], é citado entre os fatores externos de qualidade de software, a reusabilidade. Ela influencia todos os outros aspectos da qualidade (correção, robustez, extensibilidade, compatibilidade, eficiência, portabilidade, facilidade de uso, funcionalidade, conveniência), mesmo que indiretamente. Isso porque solucionar o problema da reusabilidade essencialmente significa que menos software precisará ser escrito, e portanto mais esforço pode ser empregado, dentro do mesmo custo total, na melhoria de outros fatores.

O próprio processo de manutenção de um sistema também não deixa de ser uma forma de reuso (no caso, reuso da versão anterior do sistema). E a experiência na manutenção de sistemas pode ser aproveitada para melhorar os processos de reuso de toda a empresa. [Tracz 1988] cita ganhos de até 90% na manutenção de sistemas criados utilizando técnicas de reuso.

2.3 Problemas do reuso

Reuso é uma estratégia de grande potencial; no entanto, é um potencial até hoje não totalmente explorado. Existem problemas em várias áreas que impedem a realização completa da promessa do reuso, a saber: técnicos, comportamentais, psicológicos, gerenciais e econômicos. [Yongbeom & Stohr 1998] fornece uma tabela (tabela 2.1) de aspectos relacionados a reuso, juntamente com problemas relacionados com cada um.

Tabela 2.1. Estrutura para pesquisa de reuso de software

Categoria	Aspecto pesquisado	Impedimentos ao reuso
Gerais	Definição e escopo	Falta de terminologia para a descrição de conceitos
	Aspectos econômicos	Investimento necessário para promover reuso de software; falta de modelo econômico para explicitar os benefícios e custos do reuso de software
Técnicos	Processo de reuso	Falta de metodologia para criar e implementar o reuso de software
	Tecnologias de reuso	Falta de recursos de software reutilizáveis e

		confiáveis; falta de tecnologias e técnicas para apoiar o reuso de software
Não-técnicos	Aspectos comportamentais	Falta de comprometimento, encorajamento, treinamento e recompensas pelo reuso de software; síndrome NIH (não inventado aqui)
	Aspectos organizacionais	Falta de apoio organizacional para instituir o reuso de software; dificuldade para medir os ganhos do reuso
	Aspectos legais e contratuais	Problemas com direitos de propriedade intelectual e contratuais

2.4 Práticas de reuso na engenharia de software

Scott Ambler, John Malbone e Michael Vizados, no capítulo intitulado “The Strategic Reuse Discipline” do livro [Ambler et al. 2005], fazem um levantamento das principais categorias de reuso praticadas atualmente. São elas: reuso de arquitetura, de padrões, de *frameworks*, reuso de artefatos, de módulos, de modelos e de código. As principais práticas de reuso, na atualidade, são: reuso de artefatos, de código, de modelos, componentes, padrões, *frameworks* e *assets*. Estas práticas serão descritas a seguir.

A descrição de cada uma destas categorias de reuso, com algumas considerações sobre a sua aplicação no processo de desenvolvimento, é apresentada na tabela 2.2.

Tabela 2.2: Categorias de Reuso.
Fonte: Ambler et al. 2005.

<u>Categoria</u>	<u>Descrição</u>	<u>Considerações</u>
<u>Reuso Arquitetado</u>	<u>A identificação, desenvolvimento e suporte de artefatos reusáveis de larga-escala via uma arquitetura corporativa. A arquitetura corporativa pode definir componentes de domínio reusáveis, coleções de classes de domínio/negócio relacionadas que funcionem juntas de forma a suportar um conjunto coeso de responsabilidades, ou serviços de domínio, agrupadas em um pacote coeso de uma coleção de serviços.</u>	<u>Fornecer um nível de reuso entre aplicações muito elevado.</u> <u>Requer uma abordagem sofisticada para definir a arquitetura corporativa.</u> <u>Requer uma equipe de reuso para dar suporte e evolução dos artefatos reusáveis ao longo do tempo.</u>
<u>Reuso de Padrões</u>	<u>O uso de abordagens documentadas para resolver problemas em comum dentro de um contexto específico. Com o reuso de padrões, o desenvolvedor não está reusando código, ao invés disso, o desenvolvedor está usando os conceitos que estão por trás do código. Padrões podem ser de múltiplos níveis – análise, projeto e arquitetura são os mais comuns. O site de Ward Cunningham (www.c2.com) é uma fonte de padrões bastante útil na Web.</u>	<u>Podem ser implementados em múltiplas linguagens de programação e plataformas.</u> <u>Aumenta a manutenibilidade e a capacidade da aplicação de se tornar melhor por estar usando abordagens em comum para problemas que são reconhecíveis por desenvolvedores experientes.</u> <u>Padrões não provêm uma solução imediata – o desenvolvedor ainda tem que aplicá-lo corretamente e implementar o código.</u>

Categoria	Descrição	Considerações
<u>Reuso de frameworks</u>	<p>Uso de coleções de classes que implementam em conjunto a funcionalidade básica de um domínio técnico ou de negócios.</p> <p><i>Frameworks</i> horizontais, tais como <i>frameworks</i> de segurança ou <i>frameworks</i> de interface com o usuário (como a biblioteca de classes <i>Java Foundation Class – JFC</i>), e <i>frameworks</i> verticais, tais como um <i>framework</i> de serviços financeiros, são comuns.</p>	<p>Fornecem um bom ponto-de-partida para desenvolver uma solução para um problema de domínio.</p> <p>São, seguidas vezes, uma boa opção para revisar as melhores práticas da indústria.</p> <p><i>Frameworks</i>, freqüentemente, encapsulam lógica complexa que levaria anos para ser desenvolvida desde o início.</p> <p>A complexidade dos <i>frameworks</i> torna-os difíceis de serem dominados, demandando longos processos de aprendizagem por causa de uma íngreme curva de aprendizagem.</p> <p><i>Frameworks</i> geralmente são de plataformas específicas, amarrando o desenvolvedor a um único fornecedor e, conseqüentemente, aumentando o risco do projeto.</p> <p>Embora os <i>frameworks</i> implementem 80% da lógica necessária, são geralmente os 80% mais fáceis. A parte difícil, que é a lógica de negócio e os processos que são únicos de uma organização, ainda precisa ser feita pelos desenvolvedores.</p> <p><i>Frameworks</i> raramente funcionam juntos, pois são comumente construídos sobre bases arquiteturais diferentes. A menos que eles venham de um mesmo fornecedor ou consórcio de fornecedores.</p> <p><i>Frameworks</i> verticais freqüentemente exigem que se modifique o negócio da organização para adequar-se a eles.</p>
<u>Reuso de artefatos</u>	<p>O uso de artefatos de desenvolvimento previamente criados – casos de uso, documentos-padrão, modelos de domínio específicos, procedimentos e recomendações, e até mesmo outras aplicações tais como aplicações comerciais de “prateleira” (<i>commercial-off-the-shelf - COTS</i>) – para iniciar rapidamente um novo projeto.</p> <p>Algumas vezes, o desenvolvedor reusa o artefato tal como ele é e, em outras, o artefato é utilizado como um exemplo para o desenvolvedor ter uma idéia de como continuar com o projeto.</p>	<p>Promove consistência entre projetos.</p> <p>Com freqüência, as pessoas podem comprar ou baixar da Internet muitos artefatos. Padrões de codificação de linguagens de programação estão amplamente disponíveis. Exemplos de modelos, tais como modelos de casos de uso, também são muito fáceis de serem encontrados.</p> <p>É muitas vezes percebido pelos programadores radicais como “reuso <i>overhead</i>” – o programador simplesmente muda os documentos de uma escrivania para outra.</p>

Categoria	Descrição	Considerações
<u>Reuso de módulos</u>	<u>O uso de “módulos” – componentes, serviços ou bibliotecas de código - pré-construídos e completamente encapsulados para o desenvolvimento de aplicações. Os módulos são auto-suficientes e encapsulam somente um conceito. O reuso de módulos difere do reuso de código pelo fato de que no reuso de módulo não se tem acesso ao código fonte.</u>	<p><u>Oferece um escopo maior de reusabilidade do que o reuso de código porque o desenvolvedor, literalmente, “conecta” os módulos e eles saem funcionando.</u></p> <p><u>O uso em larga escala de determinadas plataformas e padrões provê um mercado grande o suficiente para que fornecedores criem e vendam módulos aos desenvolvedores a um custo baixo. Como os módulos são, geralmente, pequenos, acaba acarretando que os desenvolvedores têm que manter uma grande biblioteca deles.</u></p> <p><u>É comum que o desenvolvedor não possa modificar um módulo, embora esta situação esteja mudando com o crescimento de popularidade da programação orientada a aspectos (POA), e, às vezes, as cláusulas de licenciamento o tornam difícil de ser utilizado.</u></p>
<u>Reuso de modelos</u>	<u>A prática de usar um conjunto comum de layouts para artefatos-chave de desenvolvimento - documentos, modelos e códigos-fonte - na organização.</u>	<p><u>Aumentam a consistência e a qualidade dos artefatos de desenvolvimento.</u></p> <p><u>As pessoas têm uma tendência a modificar modelos para o seu próprio uso e não compartilhar essas alterações com os colegas.</u></p>
<u>Reuso de código</u>	<u>O reuso de código-fonte dentro de seções de uma aplicação e, potencialmente, através de múltiplas aplicações. No melhor caso, o reuso de código é alcançado compartilhando-se classes e coleções de funções e rotinas em comum. No pior caso, o reuso de código se faz copiando e depois modificando o código existente, causando um pesadelo para o pessoal da manutenção.</u>	<p><u>Reduz a quantidade real de código que o desenvolvedor precisa escrever, diminuindo potencialmente os esforços tanto do desenvolvimento quanto da manutenção.</u></p> <p><u>Códigos genéricos/reusáveis são inerentemente mais complexos do que códigos feitos para um propósito único, tornando-os mais difíceis de serem desenvolvidos e mantidos (do ponto-de-vista de um único projeto).</u></p> <p><u>O escopo de atuação é limitado à programação.</u></p> <p><u>Pode aumentar o acoplamento dentro da aplicação.</u></p>

A figura 2.1 apresenta os diferentes graus de benefícios que são obtidos em cada categoria de reuso (representados pela seta da esquerda), bem como os diferentes níveis de dificuldade para incorporar a prática de determinada categoria de reuso em um processo de desenvolvimento (representados pela seta da direita). O reuso de módulos, por exemplo, geralmente dá mais produtividade do que o reuso de código. A prática de reuso de código e de modelos é relativamente mais fácil de ser incorporada, pois as equipes de desenvolvimento somente precisam de um instrumento para localizar o artefato reusável para poder trabalhar com ele. Já a prática de reuso de um *framework* é bem mais difícil de ser absorvida, pois as equipes de desenvolvimento, além de conhecer uma forma de obtê-lo, precisam aprender sobre este *framework*.

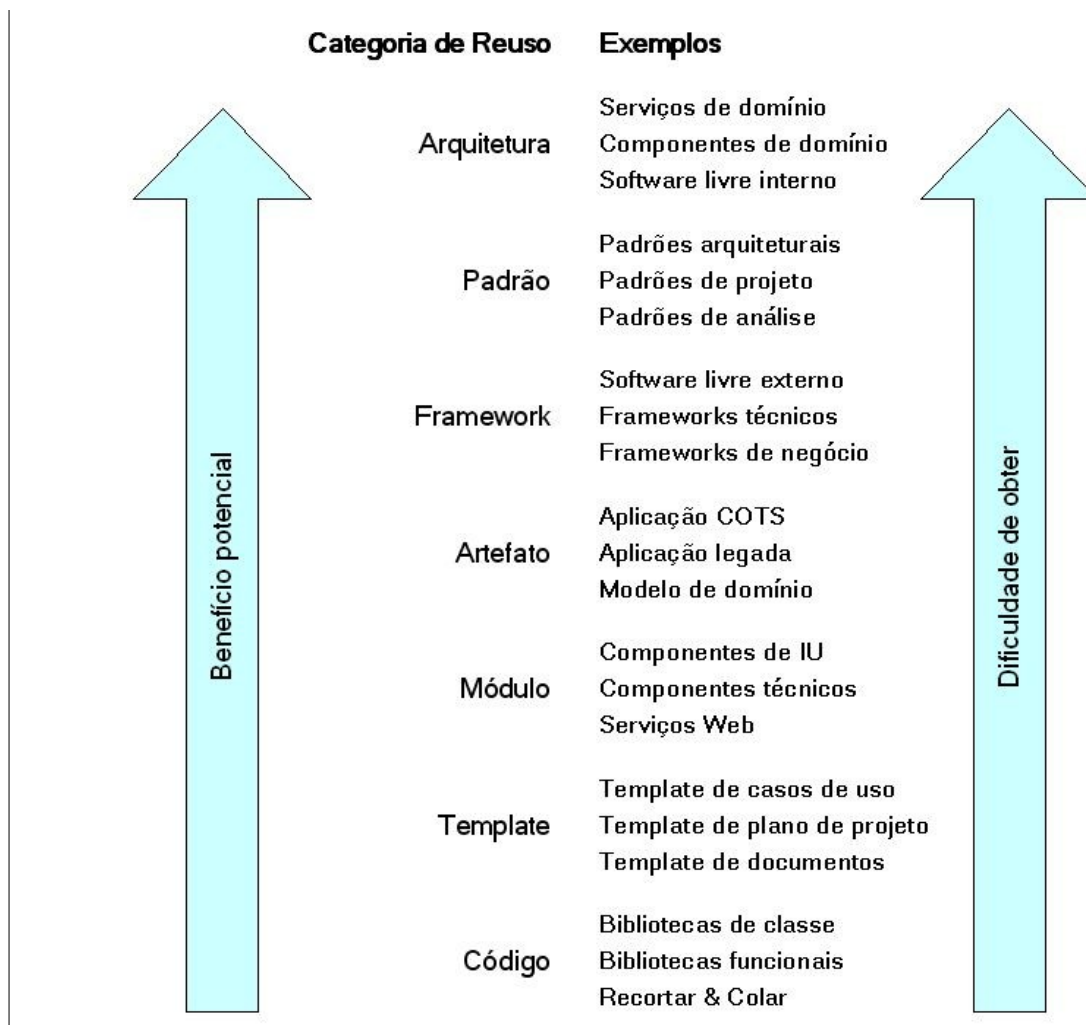


Figura 2.1: Categorias de reuso.
Fonte: Ambler et al. 2005.

2.5 Assets: Uma especificação de artefatos reusáveis

Segundo o OMG, um *asset* é uma coleção de artefatos reusáveis que fornecem uma solução para um problema em um dado contexto [OMG 2005]. Esses artefatos reusáveis podem ser quaisquer artefatos utilizados no processo de desenvolvimento de um software (documentos de requisitos, modelos, arquivos de código-fonte, casos de testes, descritores de implantação, etc). Em geral, um artefato está associado a um arquivo.

2.5.1 Reuso de artefatos

A reutilização de artefatos refere-se ao uso de artefatos de desenvolvimento previamente criados — casos de uso, documentos padrão, modelos específicos de domínio, procedimentos e diretrizes, e outras aplicações — para dar o chute inicial em um novo projeto. Há diversos níveis de reutilização de artefatos, indo dos 100% de reutilização onde o desenvolvedor toma um artefato tal qual ele é e o utiliza em um novo projeto. Por exemplo, documentos padrão tais como padronização de código e de interface são artefatos valiosos para reutilização entre projetos, assim como os documentos de notação de modelagem e os de resumo de metodologia.

A reutilização de artefatos promove consistência entre projetos e reduz a carga de gerenciamento para cada novo projeto.

Uma outra vantagem é que freqüentemente podem ser adquiridos vários artefatos (ou encontrá-los disponíveis on-line): padrões de interface de usuário são comuns para muitas plataformas, padrões de codificação para as principais linguagens estão geralmente disponíveis e metodologias padronizadas de orientação a objetos e notações de modelagem têm estado disponíveis há anos.

2.5.2 Reuso de código

Reuso de código é o tipo mais comum de reutilização e refere-se à reutilização de código-fonte em partes de uma aplicação. No melhor caso, o reuso de código é alcançado compartilhando-se classes e coleções de funções e rotinas comuns. No pior caso, a reutilização de código se faz copiando e depois modificando o código existente. Uma lamentável realidade da nossa indústria é que copiar é, geralmente, a única forma de reutilização praticada pelos desenvolvedores.

Um aspecto chave da reutilização de código é de que é necessário ter acesso ao código-fonte. Quando necessário, o próprio desenvolvedor o modifica ou demanda para alguém modificá-lo pra ele. Isso ao mesmo tempo é bom e é ruim. Ao olhar o código, o desenvolvedor pode determinar se ele quer ou não reutilizá-lo. Ao mesmo tempo, liberando o código totalmente para alguém, o desenvolvedor original pode ficar menos motivado a documentá-lo adequadamente, aumentando o tempo para entendê-lo e, conseqüentemente, diminuindo o benefício do reuso.

A principal vantagem do reuso de código é que ele reduz a quantidade real de código que se precisa escrever, diminuindo potencialmente os custos, tanto do desenvolvimento quanto da manutenção. As desvantagens são que o escopo do efeito é limitado à programação e geralmente aumenta o acoplamento dentro da aplicação.

2.5.3 Reuso de Modelos (*Templates*)

O reuso de modelos é tipicamente uma forma de reutilização de documentação. Refere-se à prática de usar um conjunto comum de *layouts* para artefatos-chaves de desenvolvimento - documentos, modelos e código-fonte - na sua organização. Por exemplo, é muito comum que as organizações adotem modelos comuns de documentação para casos de uso, cronogramas de desenvolvimento, solicitações de mudança, requisitos de usuário, arquivos de classes e cabeçalhos de documentação de métodos.

A principal vantagem dos modelos de documentação é que eles aumentam a consistência e a qualidade dos artefatos de desenvolvimento. A principal desvantagem é que os desenvolvedores têm uma tendência a modificar modelos para seu próprio uso e não compartilham essas alterações com os colegas.

Para obter melhores resultados com modelos, é preciso tornar fácil para os desenvolvedores trabalharem com eles. Existem implementações de modelos tão simples como modelos de documentos de um processador de textos, quanto como complexos bancos de dados compartilhados por todos os desenvolvedores. A organização também precisa providenciar treinamento sobre como e quando usar esses modelos, de forma que qualquer um possa usá-los consistente e corretamente.

2.5.4 Componentes

O reuso de componentes refere-se ao uso de componentes pré-construídos e totalmente encapsulados para o desenvolvimento das aplicações. Os componentes são tipicamente auto-suficientes e encapsulam um único conceito. A reutilização de componentes difere da reutilização de código pelo fato de que não se tem acesso ao código fonte. Difere-se da reutilização de herança porque não faz criação de subclasses. Exemplos comuns de componentes são os Java Beans e componentes ActiveX.

Há muitas vantagens na reutilização de componentes. Primeiro: ela oferece um escopo maior de reusabilidade do que o da reutilização de código ou de herança, porque os componentes são auto-suficientes — o desenvolvedor, literalmente, os “conecta” e eles fazem o trabalho. Segundo: a utilização ampla de plataformas comuns, como o sistema operacional Win32 e a JVM (Java Virtual Machine), provê um mercado amplo o bastante para que terceiros criem e vendam componentes a baixo custo.

A principal desvantagem da reutilização de componentes é que os componentes são pequenos e encapsulam um único conceito o que acaba acarretando em manter uma grande biblioteca deles.

O caminho mais fácil para o trabalho com componentes é começar com objetos (*widgets*) da interface de usuário — barras de deslocamento, componentes gráficos e botões gráficos (para nomear alguns). Além disso, existem componentes que encapsulam recursos do sistema operacional, tal como acesso à rede, ou que encapsulam recursos de persistência, tais como componentes que acessam bancos de dados relacionais.

Se a organização decide construir seus próprios componentes, deve ter o cuidado de que eles façam somente uma coisa. Por exemplo, um componente de interface de usuário para edição de endereços postais é bastante reutilizável por que pode ser usado em várias telas de edição. Entretanto, um componente que edita endereços postais, endereços de e-mail e um número de telefone já não é tão reutilizável — não há muitas situações onde se queira todos esses três recursos simultaneamente. Em vez disso, é melhor construir três componentes reutilizáveis e usar cada um quando necessário. Quando um componente encapsula apenas um conceito, é um componente coeso.

Como pano-de-fundo da atividade de reuso de componentes, está a discussão e definição das estratégias de reuso mais largamente encontradas na literatura, a saber: reuso de caixa-branca, reuso de caixa-preta com desenvolvimento de componente “*in-house*” e reuso de caixa-preta com componentes encontrados no mercado [Ravichandran & Rothenberger et al. 2003].

2.5.5 Padrões (*Patterns*)

Os conceitos de *padrões (patterns)* e de *linguagens de padrões* são derivados da Arquitetura e foram propostos por Christopher Alexander et al. em [Alexander et al. 1977]. Entretanto, os padrões começaram a se tornar populares na área de TI com a larga aceitação do livro “Padrões de Projeto — Soluções reutilizáveis de Software Orientado a Objetos” de Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (conhecidos como GoF — Gang of Four) [Gamma et al. 2000].

Atualmente, a comunidade de software está usando padrões de forma ostensiva na arquitetura de software e no projeto de software.

Um padrão descreve uma solução provada para um problema em um certo contexto de projeto. Um padrão tem campos definidos: Contexto, Problema, Forças e Solução. Um padrão é usado em um certo Contexto de projeto e considera um Problema de projeto recorrente neste Contexto. Ele foca-se nas Forças as quais o projetista se confronta antes de descrever a Solução — uma abordagem proposta para a situação que resolve as tensões entre as forças.

Como exemplo, considere o padrão de Alexander — “Um Lugar para Esperar” [Alexander et al. 1977]. O contexto é qualquer situação onde as pessoas estão esperando por algo, tal como uma sala de espera de um consultório médico. Duas forças conflitam: (a) pacientes devem estar presentes quando o médico está disponível, mas (b) o momento deste evento é incerto, levando a uma situação de ansiedade. Uma solução sugerida é de ocupar as pessoas que estão esperando. Um hospital criou um playground que funcionava como uma sala de espera para as crianças de forma que os jovens-pacientes se sentissem relaxados e confiantes antes das suas consultas.

Um padrão em específico pode contribuir para melhorar o reuso, mas os maiores ganhos se obtêm quando os padrões são cuidadosamente combinados. Uma vez que a solução para um padrão foi aplicada, um novo contexto surge no qual problemas mais detalhados demandam solução. Outros padrões podem ser descobertos para capturar o processo de problema-solução inerente deste novo contexto.

Mahemoff reforça que uma *linguagem de padrões* é constituída quando uma coleção de padrões é arranjada em uma rede de padrões interdependentes, especialmente onde padrões de mais alto nível rendem contextos os quais são resolvidos por padrões mais detalhados [Mahemoff & Johnston 2001].

2.5.6 Frameworks

O reuso de *frameworks* refere-se ao uso de coleções de classes que implementam em conjunto a funcionalidade básica de um domínio técnico ou de negócios. Os desenvolvedores usam *frameworks* como o alicerce a partir do qual eles constroem uma aplicação; os 80% comuns já estão no lugar, eles apenas necessitam acrescentar os restantes 20% específicos da aplicação.

Frameworks que implementam os componentes básicos de uma GUI são muito comuns. Há *frameworks* para seguros, recursos humanos, manufatura, bancos e comércio eletrônico. A reutilização de *frameworks* representa um alto nível de reutilização no nível do domínio.

Os *frameworks* fornecem uma solução inicial para um domínio de problema e freqüentemente encapsulam uma lógica complexa que levaria anos para ser desenvolvida desde o rascunho.

Infelizmente, a reutilização de *frameworks* sofre de diversas desvantagens. A complexidade dos *frameworks* torna-os difíceis de serem dominados, exigindo um longo processo de aprendizagem por parte dos desenvolvedores. *Frameworks* geralmente são de plataformas específicas, amarrando o desenvolvedor a um único fornecedor e, conseqüentemente, aumenta o risco da sua aplicação.

Embora os *frameworks* implementem 80% da lógica necessária, são geralmente os 80% mais fáceis. A parte difícil que é a lógica de negócio e os processos que são únicos de uma organização, ainda precisa ser feita.

~~*Frameworks* raramente trabalham juntos, a menos que eles venham de um mesmo fornecedor ou consórcio de fornecedores. Eles sempre exigem que se modifique o negócio da organização para adequar-se ao *framework* em vez de outro caminho.~~

2.5.7 *Assets*

~~Segundo o OMG, um *asset* é uma coleção de artefatos reusáveis que fornecem uma solução para um problema em um dado contexto [OMG 2005].~~

~~Um *asset* é composto por um conjunto de artefatos reusáveis que fornecem a solução para o problema apresentado pelo *asset*. Esses artefatos reusáveis podem ser quaisquer artefatos utilizados no processo de desenvolvimento de um software (documentos de requisitos, modelos, arquivos de código-fonte, casos de testes, descrições de implantação, etc). Em geral, um artefato está associado a um arquivo.~~

Um *asset* pode ter pontos de variabilidade que permitem aos usuários customizá-lo através da configuração de vários parâmetros. Um ponto de variabilidade é uma localização no *asset* que pode ter um valor fornecido ou customizado pelo consumidor do *asset*. *Assets* têm regras de uso, as quais são instruções que descrevem como o *asset* deve ser usado. Estas regras diminuem o tempo que os desenvolvedores levam para descobrir, analisar, consumir e testar o *asset*. Adicionalmente, um *asset* descreve o contexto de desenvolvimento e de negócio no qual ele pode e deveria ser reusado.

Um *asset* pode ser um serviço, um *pattern*, um componente ou outro elemento que forneça uma solução.

Note que a definição de um *asset* é similar à de um padrão (*pattern*). Isto se deve ao fato de que os conceitos fundamentais por trás de ambos utilizam o trinômio contexto-problema-solução, mas a diferença está nos detalhes. Um *asset* é um conceito mais genérico que um padrão. Estritamente falando, os pontos de variabilidade de um *asset* estão no nível dos artefatos, ao passo que um padrão terá parâmetros e tipos de pontos de variabilidade aplicáveis ao padrão como um todo e não necessariamente a um artefato em particular.

A definição geral do *asset* apresentada acima é refinada para vários tipos de *assets* de software. Um tipo específico de *asset* pode especificar os artefatos que devem estar no *asset* e podem declarar um contexto específico tal como um contexto de desenvolvimento ou um contexto de execução para o qual o *asset* é relevante.

Existem três dimensões que descrevem *assets* reusáveis: granularidade, variabilidade e articulação. Estas dimensões são ilustradas na figura 2.21.

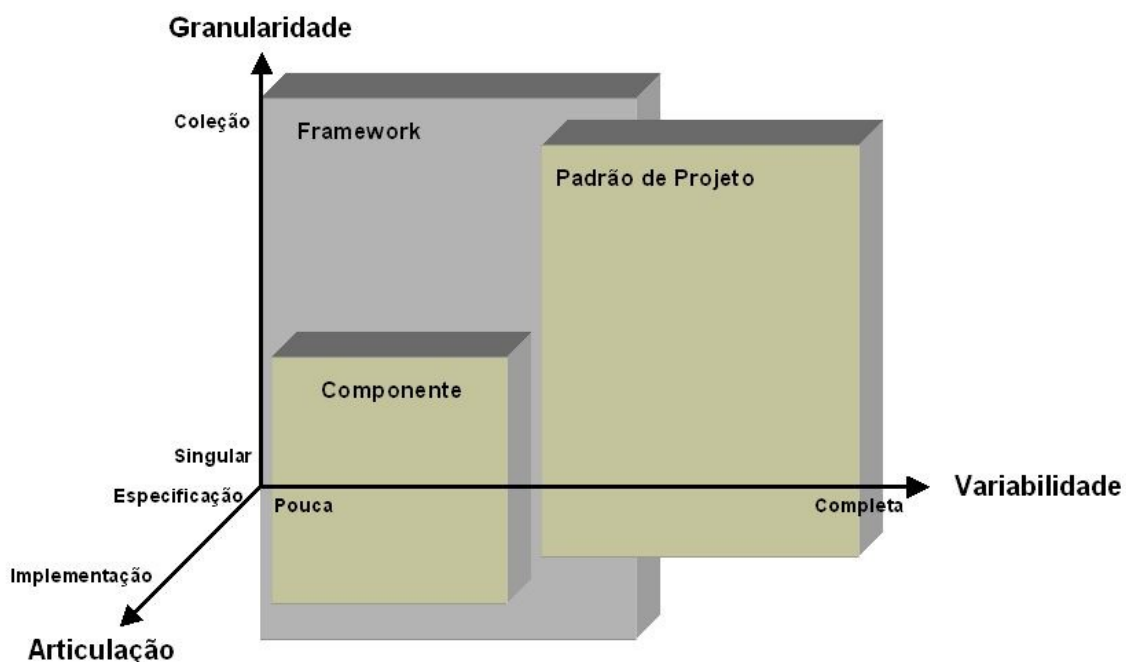


Figura 2.21: Dimensões de *assets* reusáveis.
Fonte: OMG 2005

- Granularidade.

A granularidade de um *asset* descreve quantos problemas ou alternativas de solução são endereçados por um *asset*. Os *assets* mais simples oferecem uma solução singular para um único e bem definido problema. Na medida que a granularidade aumenta, o *asset* endereça múltiplos problemas e/ou pode oferecer soluções alternativas a esses problemas. O produtor do *asset* deve decidir se coloca todos os artefatos em um único *asset* ou divide em vários *assets* menores mas mais simples.

- Variabilidade.

Refere-se à quão customizável o *asset* deve ser. Num extremo do espectro, um *asset* é invariável, ou seja, ele não pode ser alterado. Este é o caso de *assets* que são componentes binários. *Assets* neste extremo do espectro são chamados de *assets* caixa-preta (*black-box assets*) devido ao fato de que sua parte interna não pode ser vista e não é modificável.

No outro extremo do espectro estão os *assets* caixa-branca (*white-box assets*). Estes *assets* são criados com a expectativa de que os consumidores do *asset* irão editá-los de forma a alterar sua implementação. Tipicamente, os *assets* caixa-branca também incluem artefatos de desenvolvimento tais como requisitos, modelos, arquivos de “build”, etc.

Existem duas outras variações que estão no meio-termo: são os *assets* caixa-clara (*clear-box assets*) e os *assets* caixa-cinza (*grey-box assets*). *Assets* caixa-clara expõem os detalhes de implementação (via modelos, fragmentos de código ou outra documentação), entretanto eles não podem ser modificados. Estes detalhes são expostos somente para ajudar o consumidor do *asset* a entender melhor o funcionamento do *asset*.

de forma que possa usá-lo mais eficientemente. *Assets* caixa-cinza expõem e permitem modificação somente em um subconjunto de artefatos do *asset*, usualmente através de parâmetros do *asset*.

- Articulação.

A dimensão de articulação descreve o grau de completeza dos artefatos em fornecer a solução. *Assets* cujos artefatos especificam uma solução mas não a implementam têm um baixo grau de articulação, ao passo que *assets* cujos artefatos especificam e implementam a solução juntamente com documentos de suporte (tais como requisitos, casos de uso, artefatos de teste, e assim por diante) têm um alto grau de articulação.

Os *assets* reusáveis devem ser documentados e disponibilizados em um repositório que possa ser utilizado pelas equipes de projeto. Para documentar, empacotar e catalogar os diversos *assets*, o OMG propôs a especificação RAS (*Reusable Asset Specification*) [OMG 2005]. Usando RAS, os desenvolvedores podem empacotar cada *asset* com um envelope de meta-dados com as informações básicas sobre o *asset*, denominado de *Core RAS*, cuja estrutura é apresentada na figura 2.32.



Figura 2.32: Principais seções do Core RAS.

Fonte: OMG 2005

A figura 2.3 mostra que um *asset* RAS tem uma seção *Classificação* para facilitar a pesquisa e a navegação por tópicos (*browsing*). Esta seção pode incluir pares de descritores do tipo nome/valor e uma declaração dos contextos, tais como um contexto específico de domínio, de desenvolvimento ou de implantação. A seção *Solução* é a de maior interesse, pois é nela que está a coleção de artefatos que provêem a solução.

~~A figura 2.2 mostra que um *asset* RAS tem uma seção *Classificação* para facilitar a pesquisa e a navegação por tópicos (*browsing*). Esta seção pode incluir pares de descritores do tipo nome/valor e uma declaração dos contextos, tais como um contexto específico de domínio, de desenvolvimento ou de implantação. A seção *Solução* é a de maior interesse, pois é nela que está a coleção de artefatos que provêem a solução.~~

A seção *Uso* passa orientações de como aplicar e utilizar o *asset* usando os pontos de variabilidade. Algumas dessas orientações de uso podem ser automatizadas através de scripts ou assistentes, os quais também estarão armazenados na seção *Solução* juntamente com os outros artefatos do *asset*. A seção *Assets Relacionados* define os relacionamentos do *asset* com outros *assets*.

Para suportar os vários graus de reuso, formalidade e maturidade do processo de reuso nas organizações, muitas das seções RAS são opcionais.

Enquanto o *Core RAS* contém informações genéricas sobre os *assets*, certamente que informações mais detalhadas são requeridas para especificar certos tipos de *assets*, tais como *web services*, padrões, componentes e *frameworks*. Para isso, a especificação RAS pode ser estendida e customizada através de *perfis* (*profiles*). Estes perfis preservam e estendem a descrição núcleo do RAS apresentada acima. Atualmente, a OMG define três perfis: *Default*, *Default Component* e *Default Web Service*. O perfil *Default* é usado para empacotar qualquer tipo de *asset* de software enquanto que os demais são para uso com componentes e *Web services*, respectivamente.

~~Um *Core RAS* não pode ser instanciado. Portanto, um *asset* deve estar associado a um perfil. Um perfil pode estender um *Core RAS* ou outro perfil.~~

2.6 Processos de reuso

O ciclo de vida dos artefatos reusáveis compreende os processos pelos quais eles são identificados, obtidos, criados, documentados, certificados, publicados, reusados, mantidos, medidos e – finalmente - aposentados. O OMG tem trabalhado numa abordagem de desenvolvimento de software chamada desenvolvimento baseado em *assets* (*asset-based development* - *ABD*) que descreve todo esse ciclo de vida em quatro fluxos de trabalho distintos: identificação, produção, gerenciamento e consumo de *assets* reusáveis (ver figura 2.43). Estes fluxos de trabalho são congregados através de processos, ferramentas e padrões [Larsen & Wilber 2005].

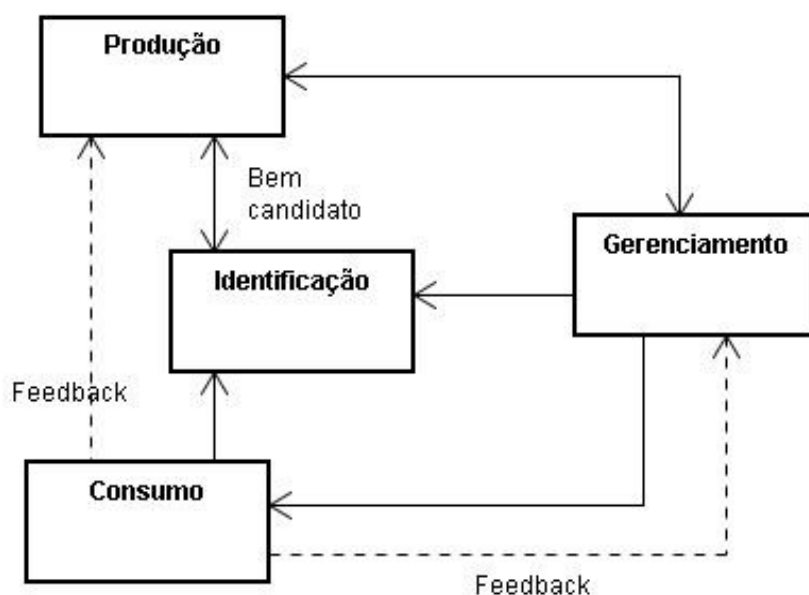


Figura 2.43: Os fluxos de trabalho do ciclo de vida de um artefato reusável.

Fonte: Larsen & Wilber 2005

O processo de reuso deve ocorrer de forma independente mas conectado (integrado) com as demais atividades do processo de desenvolvimento do sistema.

- Processo de identificação

O primeiro passo no processo de identificação é o de identificar o problema recorrente e uma solução em potencial. A pessoa responsável pela execução deste passo fará uma pesquisa por artefatos ou *assets* candidatos e, encontrando-os, os submeterá para serem preparados para reuso, empacotados como um *asset* RAS e armazenados em um repositório RAS. O objetivo é identificar todos os potenciais artefatos existentes que são úteis para o restante da organização e para serem reusados em futuros projetos de software.

- Processo de produção

Produzir um *asset*, envolve o trabalho de modelagem, codificação, teste e documentação do *asset*. O indivíduo ou equipe responsável pelo processo de produção transformará o *asset* candidato em um autêntico *asset* reusável. Neste processo, um novo *asset* poderá ser criado ou serão desenvolvidos artefatos complementares e refinados os existentes.

O *asset* deve ser concebido de forma a ser altamente consumível. Para isso, o responsável pela sua produção, deve concebê-lo levando em conta as três dimensões-chave vistas acima: variabilidade, granularidade e articulação.

Tendo sido construído e testado, o *asset* é então documentado, empacotado, classificado e disponibilizado em um repositório RAS para consumo.

- Processo de gerenciamento

O processo de gerenciamento compreende as atividades que atuam em todo o ciclo de vida de um *asset*. Este processo deve detalhar como um artefato é submetido como candidato a *asset*, como homologar e revisar *assets*, como publicá-los e como aposentá-los. Deve definir métricas de *assets*, criar esquemas de classificação, bem como estabelecer os papéis que as pessoas desempenharão nos processos e as atividades destes papéis. Este processo, também pode descrever como criar perfis RAS customizados às necessidades da organização.

A figura 2.5 apresenta um diagrama de estados para o ciclo de vida de um *asset*.
~~apresenta um diagrama de estados com a situação (status) que um asset passa no seu ciclo de vida.~~ Este diagrama foi modelado com base nas atividades definidas no ABD.

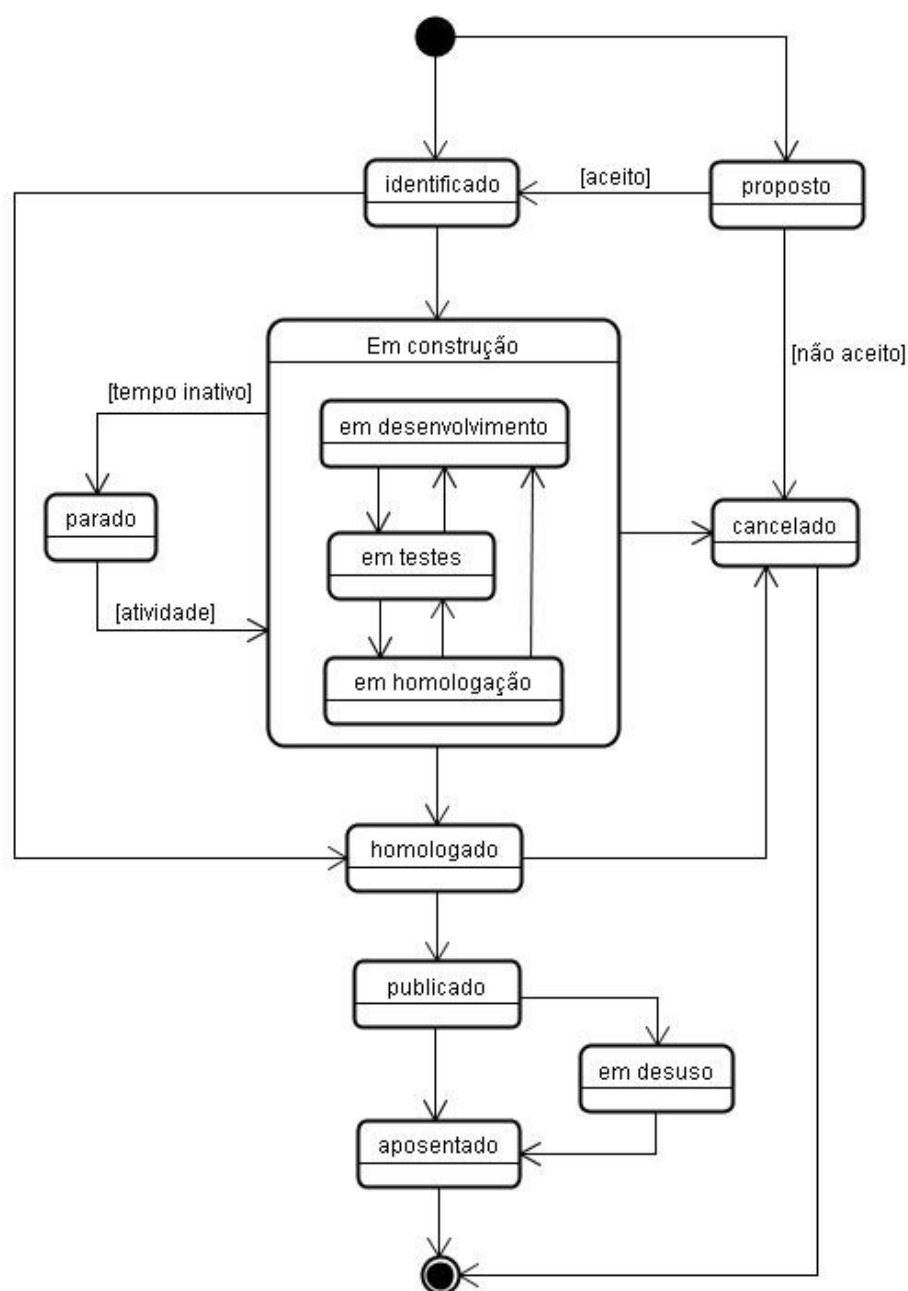


Figura 2.54: Diagrama de estados do ciclo de vida de um *asset*.

Um gerenciamento efetivo de *assets* depende predominantemente em estabelecer um processo bem definido de governança que descreva regras, políticas, atividades, estados e transições.

- Processo de consumo

No processo de consumo, uma das atividades chave é o de localização de *assets*. Pode-se localizar um *asset* através da explicitação de critérios de pesquisa (usando as palavras-chave definidas nos meta-dados do *asset*) ou através da navegação por tópicos (*browsing*). Uma vez localizado um *asset* que atenda as necessidades do projeto, o consumidor do *asset* poderá configurá-lo, usá-lo e, eventualmente, fornecer *feedback* do seu uso para os responsáveis pela produção e gerenciamento do *asset*. ~~o consumidor do *asset* poderá configurá-lo, usá-lo e, eventualmente, fornecer *feedback* do seu uso~~ na forma de relatórios de defeitos ou de requisições de mudanças.

3 DESENVOLVIMENTO DE SOFTWARE COM UML E CASOS DE USO

Considerando que nossa abordagem se baseia nos fundamentos de casos de uso, o foco principal deste capítulo – além de uma visão breve e genérica de UML na seção 3.1 – é revisar os conceitos associados a caso de uso (atores, relacionamentos entre casos de uso, níveis de casos de uso, etc) e as características de um processo de desenvolvimento dirigido por casos de uso. Estes tópicos são discutidos nas seções 3.2 e 3.3 respectivamente. A última seção deste capítulo faz uma revisão das principais abordagens para reuso de casos de uso.

3.1 O uso da UML para a modelagem de sistemas

A UML é uma linguagem visual universal para modelagem de sistemas orientados a objetos (OO). A UML é a linguagem para especificação, visualização, construção e documentação de artefatos de sistemas de software [OMG 2003].

Segundo os autores da UML, um sistema pode ser descrito por cinco visões interdependentes, onde cada visão enfatiza um aspecto diferente do sistema [Booch et al. 2000]. Estas visões são:

- Visão de Casos de Uso: abrange os casos de uso que descrevem o comportamento do sistema conforme é visto pelos usuários finais, analistas e pessoal de teste, ou seja, descreve o sistema do ponto de vista externo como um conjunto de interações entre o sistema e os agentes externos ao sistema.
- Visão de Projeto: abrange as classes, interface e colaborações que formam o vocabulário do problema e de sua solução.
- Visão de Implementação: abrange os componentes e os arquivos utilizados para a montagem e fornecimento do sistema físico, bem como o gerenciamento de versões de componentes do sistema.
- Visão de Implantação: abrange os nós que formam a topologia de hardware em que o sistema é executado. Essa visão direciona principalmente a distribuição, o fornecimento e a instalação das partes físicas do sistema.
- Visão de Processo: esta visão enfatiza as características de concorrência (paralelismo), sincronização e desempenho do sistema.

Cada uma destas visões é modelada e documentada através de artefatos gráficos chamados de *diagramas da UML*. Estes diagramas são: diagrama de casos de uso, diagrama de classes, diagrama de objetos, diagrama de transições de estados, diagrama de atividades, diagrama de sequência, diagrama de colaborações, diagrama de componentes e diagrama de implantação.

A definição, especificação e aplicação de cada um destes diagramas estão fora do escopo deste trabalho (para maiores informações, sugere-se a leitura da especificação oficial da UML pelo OMG em [OMG 2003] e do livro “UML. Guia do usuário” [Booch et al. 2000]). Entretanto, os conceitos e a especificação de casos de uso serão revisados na próxima seção por ser o caso de uso um instrumento-chave para o projeto de interfaces com o usuário e a sua conseqüente relação com as abordagens de reuso de interfaces.

3.2 Casos de uso: fundamentos

Nesta seção são revistos os conceitos de caso de uso, os elementos que os compõem e a estrutura básica da descrição de um caso de uso.

3.2.1 Conceito

Segundo a UML, um caso de uso é “uma descrição de um conjunto de seqüências de ações, inclusive variantes, que um sistema executa para produzir um resultado de valor observável por um ator” [Booch et al. 2000]. O “resultado de valor observável” é aquele produzido pelo caso de uso ao tentar alcançar o objetivo do ator. Esse resultado pode ser de sucesso ou de falha, dependendo se o caso de uso conseguiu atingir o objetivo ou não.

Casos de uso contêm narrativas que mostram como os atores interagem com o sistema para atingir um único objetivo. Essas narrativas descrevem a interação que se dá pela troca (envio e/ou recebimento) de informações entre o sistema e ao menos um ator de forma que o objetivo do ator seja alcançado [Cockburn 2005].

A idéia central dos casos de uso é de descrever o que um sistema faz, sem especificar como isso é feito, ou seja, um caso de uso captura o comportamento pretendido pelo sistema sem, entretanto, especificar como este comportamento é implementado.

Casos de uso que são completamente desprovidos de características tecnológicas são chamados de **casos de uso essenciais**. Por outro lado, os casos de uso que mencionam elementos de tecnologia na descrição das interações são chamados de **casos de uso reais**.

3.2.1.1 Casos de Uso Essenciais (de Constantine)

Casos de uso essenciais foram propostos por Larry Constantine e Lucy Lockwood em [Constantine & Lockwood 1999] com o objetivo de fazer com que a descrição dos casos de uso se concentre apenas na essência da interação para atingir o objetivo do caso de uso. Essa essência se traduz em se focalizar na descrição das *intenções* do usuário e na *responsabilidade* do sistema e não em ações concretas. A narrativa fica independente de elementos de tecnologia, especialmente aqueles relacionados com a IU [Larman 2004].

Um exemplo de caso de uso essencial é apresentado na tabela 3.1.

Para verificar se um caso de uso contém algum detalhe de tecnologia, [Bezerra 2002] propõe aplicar a “*regra dos 100 anos*” que consiste em perguntar, ao ler a narrativa de um caso de uso, se essa narrativa seria válida tanto há 100 anos atrás quanto daqui a 100 anos. Se for válida, então esse caso de uso foi descrito na forma essencial.

Tabela 3.1: Exemplo de um caso de uso essencial

Sacar dinheiro	
Intenção do usuário	Responsabilidade do sistema
Fornece a identificação	verifica identidade oferece opções disponíveis
solicita saque de determinada quantia	
retira o dinheiro e o recibo	fornece a quantia solicitada

3.2.1.2 Casos de Uso Reais

Os casos de uso reais descrevem concretamente os passos do fluxo de eventos, fazendo referências aos elementos tecnológicos utilizados na interação. Na prática, casos de uso concretos só podem ser descritos com a tecnologia a ser utilizada e o projeto da interface com o usuário definido.

Descrever elementos de interface e outros detalhes de implementação na descrição de casos de uso é visto por muitos autores como um erro, pois se antecipa prematuramente decisões e definições da fase de projeto (principalmente projeto de IU). Na fase de levantamento de requisitos, o analista deve estar focado na identificação do que o sistema deve fazer (requisitos funcionais) e não deve se ~~deve-se~~ ater a detalhes de implementação. Evitar colocar detalhes da IU faz com que a descrição do caso de uso fique mais clara e simples de se entender [Cockburn 2005]. A tabela 3.2 apresenta um exemplo de caso de uso real.

Tabela 3.2: Exemplo de um caso de uso real

Sacar dinheiro	
Ator	Sistema
Insere o cartão de identificação no leitor de cartões	Valida dados do cartão Apresenta tela de identificação
Digita a senha na tela de identificação e tecla <Enter>	Valida a senha Apresenta tela com menu numerado de opções

Digita o número que corresponde a operação de saque	
Digita o valor a ser sacado e tecla <Enter>	Sistema apresenta tela com o campo de valor a ser sacado
...	...

3.2.2 Elementos

O modelo de casos de uso de um sistema é composto de casos de uso, atores e de relacionamentos entre estes. A seguir, cada um destes elementos são apresentados em detalhes.

3.2.2.1 Atores

Para a UML, um ator é tudo aquilo que está fora do sistema e representa um conjunto coerente de papéis que os usuários dos casos de uso desempenham quando interagem com estes casos de uso. Tipicamente, um ator representa um papel que um ser humano, um dispositivo de hardware ou até outro sistema desempenha com o sistema [Booch et al. 2000]. Ou seja, qualquer elemento externo que interage com o sistema é denominado ator.

Atores têm objetivos. Eles interagem com um sistema, porque desejam que o sistema os ajude a atingir estes objetivos [Cockburn 2005; Larman 2004].

Um aspecto importante é ~~de~~ que um ator corresponde a um papel representado em relação ao sistema. Sendo assim, o nome de um ator deve lembrar o seu papel ao invés de quem o representa. Por exemplo: em um sistema de caixa eletrônico, o funcionário do banco pode interagir com o sistema ora como Cliente, quando realiza um saque, e ora como Funcionário quando faz a manutenção no caixa eletrônico. Neste caso, mesmo sendo a mesma pessoa que está utilizando o sistema, são definidos dois atores (Cliente e Funcionário).

Graficamente, os atores são representados por um boneco de palitos com o nome do papel que representam (figura 3.1)



Figura 3.1: Exemplos de Atores na UML

3.2.2.2 Casos de Uso

Na UML, um caso de uso é representado graficamente por uma elipse com uma sentença que representa o nome do caso de uso (figura 3.2). Geralmente, o nome do caso de uso representa o objetivo que o ator quer atingir ao executar o caso de uso.

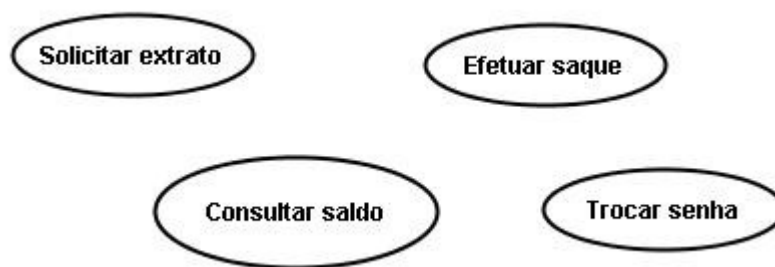


Figura 3.2: Exemplos de casos de uso na UML

3.2.2.3 Cenários

Um caso de uso, geralmente, pode ser realizado de diversas maneiras. Um cenário representa cada uma dessas maneiras. Cada cenário descreve uma situação em particular, um fluxo específico de ações percorrido pelo ator e o sistema na tentativa de atingir um objetivo.

Cenários também são chamados de *instâncias de caso de uso*. Logo, um caso de uso é uma coleção de cenários relacionados, que descrevem como os atores usam o sistema como meio de atingir um objetivo.

O fator determinante de escolha de qual cenário será executado depende particularmente do quê o ator requisita e das condições às quais o caso de uso é executado. O cenário é, então, executado até que o objetivo seja atingido ou abandonado.

Todo o caso de uso deve ter, pelo menos, um *fluxo principal* que descreva o que normalmente acontece quando um caso de uso é realizado. Este fluxo deve narrar a sequência de interações que conduzam à que o objetivo seja atingido. Geralmente, existem *fluxos alternativos* de sequências de interações que o ator escolhe como alternativa ao fluxo principal, mas que o conduzem para atingir o mesmo objetivo. Também, podem existir *fluxos de exceção* que descrevem a sequência de ações que ocorrem quando acontece algo inesperado na interação entre o ator e o sistema ou quando o sistema encontra algum obstáculo que o impeça de atingir o objetivo.

A figura 3.3 ilustra de forma esquemática os fluxos de um caso de uso.

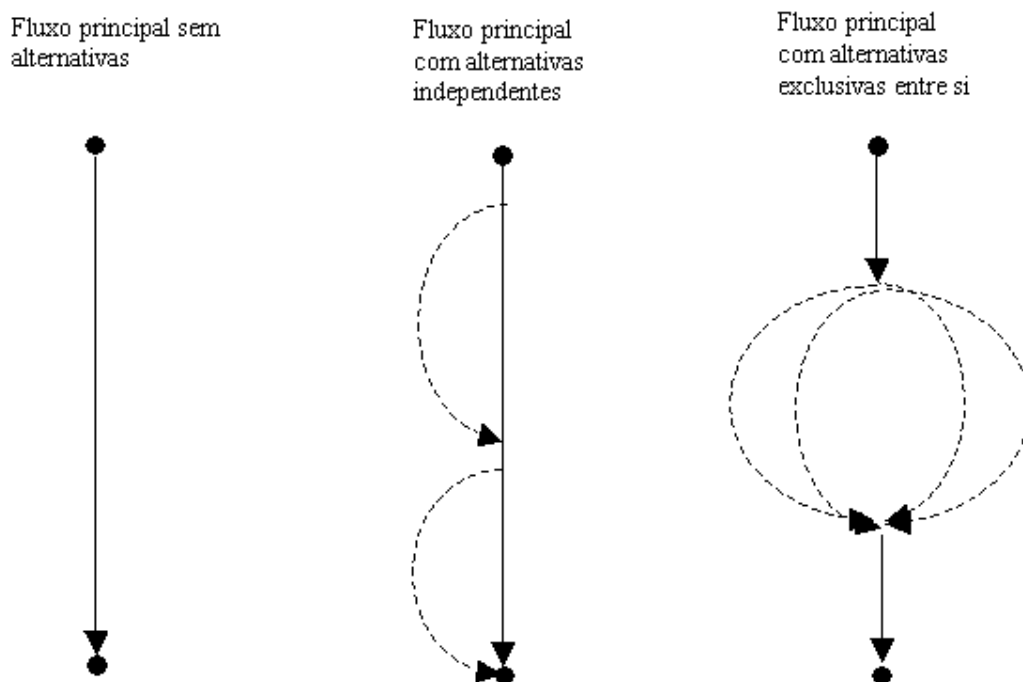


Figura 3.3: Fluxos de um caso de uso
Fonte: Bezerra 2002

3.2.2.4 Relacionamentos

Na modelagem de casos de uso, podem ocorrer situações em que um mesmo segmento de fluxo de eventos ocorre em vários casos de uso (redundância), ou que é necessário adicionar comportamento a um caso de uso sem alterar o fluxo de eventos original. Os casos de uso podem se relacionar entre si de forma a compartilhar comportamento comum ou estender esses comportamentos. A UML descreve três tipos de relacionamentos entre casos de uso: inclusão, extensão e generalização e um tipo de relacionamento entre atores e casos de uso: associação, que serão descritos a seguir.

Relacionamento de associação

O relacionamento de associação apresenta os atores que interagem (trocam informações) com um caso de uso. É representado por uma linha sólida ligando o ator e o caso de uso. A figura 3.4 exemplifica este relacionamento.

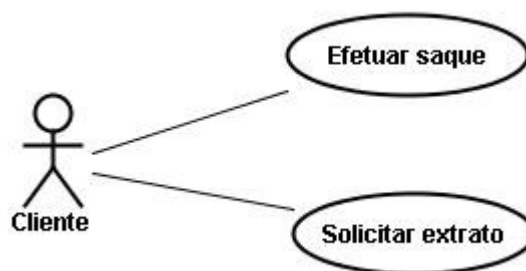


Figura 3.4: Notação da UML para relacionamento de associação.

Relacionamento de inclusão

Um relacionamento de inclusão permite que um caso de uso inclua no seu fluxo de eventos comportamentos especificados em outros casos de uso. É um bom mecanismo para capturar comportamentos comuns entre vários casos de uso, pois evita a redundância de comportamento e facilita as eventuais mudanças no modelo de casos de uso. Este mecanismo é análogo ao conceito de *rotina* nas linguagens de programação.

Um relacionamento de inclusão envolve dois casos de uso: o caso de uso incluído e o caso de uso que inclui (caso de uso base). Um caso de uso incluído é referenciado em algum ponto no fluxo de eventos do caso de uso base. Esse ponto é chamado de *ponto de inclusão*. O caso de uso incluído é executado quando o ponto de inclusão no fluxo de eventos do caso de uso base é alcançado. Após a execução do caso de uso incluído, a execução do fluxo do caso de uso base é retomada a partir do ponto de inclusão (figura 3.5). O caso de uso base não conhece detalhes do comportamento do caso de uso incluído, mas precisa conhecer o resultado da sua execução.

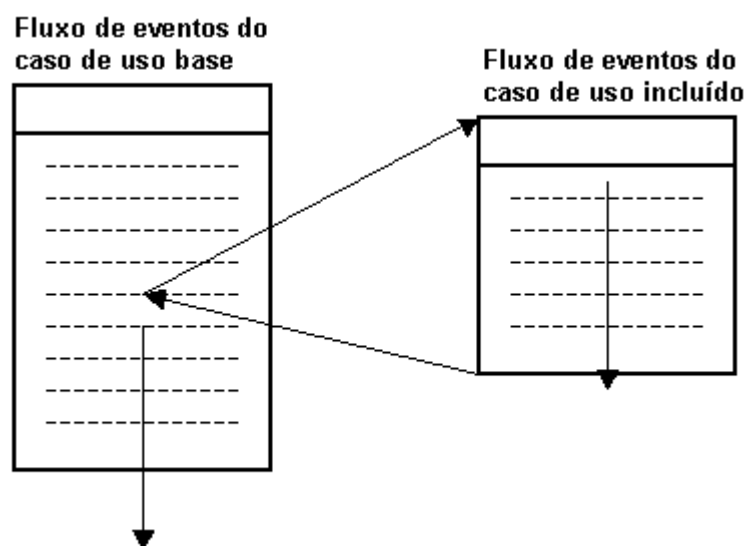


Figura 3.5: Caso de uso base que inclui comportamento de outro caso de uso.

O relacionamento de inclusão estabelece uma relação de dependência entre o caso de uso base e o caso de uso incluído, fazendo com que a execução do caso de uso incluído seja obrigatória.

A representação na UML de um relacionamento de inclusão é feita por uma seta tracejada partindo do caso de uso base para o caso de uso incluído. Esta seta é rotulada com a palavra <<include>>. A figura 3.6 ilustra um relacionamento de inclusão.

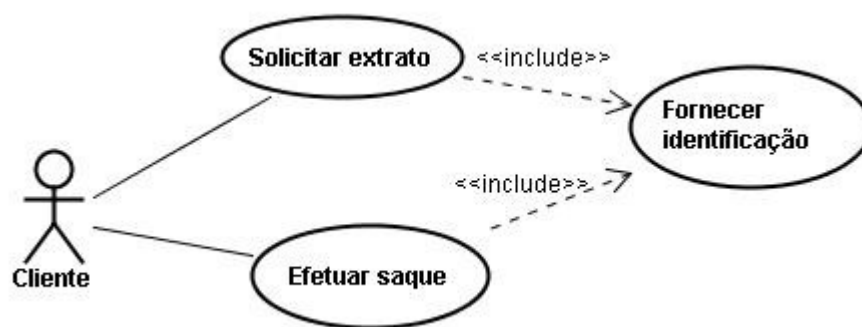


Figura 3.6: Exemplo de relacionamento de inclusão.

Relacionamento de extensão

Um relacionamento de extensão permite que o comportamento de um caso de uso seja estendido com comportamentos adicionais de outros casos de uso. Os relacionamentos de extensão permitem representar e estruturar comportamentos adicionais no caso de uso base de forma que o fluxo de eventos do caso de uso base não fique repleto de novos fluxos que poderiam ocasionar em perda de foco do fluxo de eventos principal.

O caso de uso que está sendo estendido (caso de uso base) pode conter vários *pontos de extensão*. O caso de uso base deve indicar os pontos no seu fluxo de eventos onde ele pode ser estendido e qual o nome do caso de uso que está fornecendo a extensão. Dependendo de certas condições, o caso de uso que fornece a extensão poderá ser executado a partir do ponto de extensão (figura 3.7). Após a execução do caso de uso que fornece a extensão, a execução do fluxo do caso de uso base é retomada a partir do ponto de extensão.

Opcionalmente, pode-se definir uma *condição de guarda* associada com um ponto de extensão. Se a condição de guarda é avaliada como verdadeira, então o caso de uso que fornece a extensão é executado. Caso contrário, o fluxo do caso de uso base continua normalmente como se o ponto de extensão não tivesse existido.

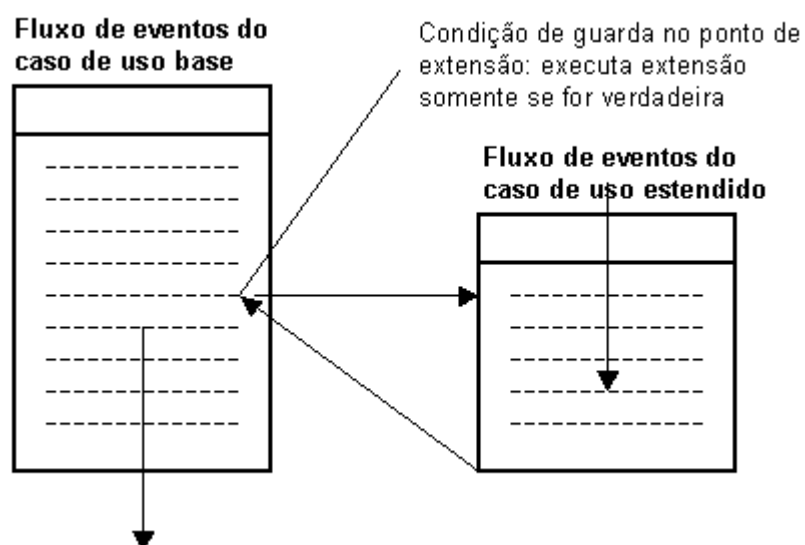


Figura 3.7: Caso de uso base com um caso de uso estendido.

Diferentemente de um relacionamento de inclusão, onde a execução do caso de uso incluído é obrigatória, um relacionamento de extensão modela a parte de um caso de uso que é considerada como opcional para que o caso de uso seja realizado.

A representação na UML de um relacionamento de extensão é feita por uma seta tracejada partindo do caso que fornece a extensão para o caso de uso base. Esta seta é rotulada com a palavra `<<extend>>`. A figura 3.8 demonstra esta notação.

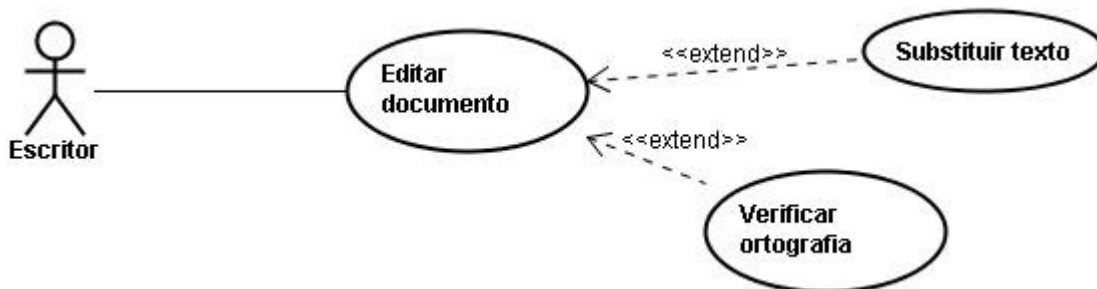


Figura 3.8: Exemplo de relacionamento de extensão.

Relacionamento de generalização

A UML descreve relacionamentos de generalização como um relacionamento entre casos de uso pai e casos de uso filhos, onde cada caso de uso filho contém todos os atributos, seqüências de interação, pontos de extensão e relacionamentos do caso de uso pai. Entretanto, o caso de uso filho representa uma forma mais especializada do comportamento genericamente definido no caso de uso pai, pois além de herdar o comportamento do caso de uso pai, pode adicionar novos comportamentos ou redefini-los sobrescrevendo alguns passos do fluxo de interação.

Resumidamente, o relacionamento de generalização de casos de uso é uma implementação do conceito de herança da Orientação a Objetos.

A representação na UML de um relacionamento de generalização é feita por uma linha cheia direcionada por uma seta aberta partindo do caso de uso filho para o caso de uso pai, conforme ilustrado na figura 3.9.



Figura 3.9: Notação da UML para relacionamento de generalização entre casos de uso.

Também podem existir relacionamentos de generalização entre atores. Atores que possuem um papel mais concreto podem ser generalizados por um ator de papel mais conceitual (figura 3.10). O ator filho pode fazer tudo o que o ator pai faz, mas o

contrário não é verdadeiro. Se um ator filho está relacionado a um caso de uso, o ator pai não pode desempenhar o papel do ator filho.

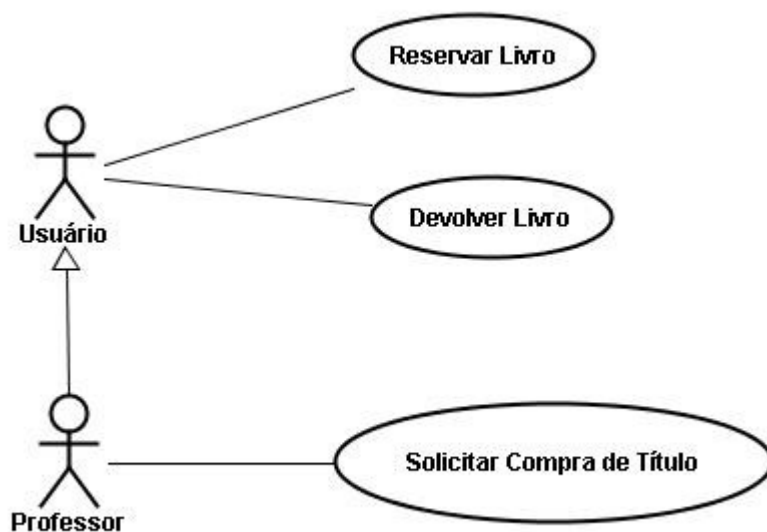


Figura 3.10: Notação da UML para relacionamento de generalização entre atores.

O propósito da generalização ou especialização entre atores é de reduzir a redundância de papéis na comunicação com os casos de uso, bem como de auxiliar na definição dos diferentes privilégios de uso do sistema.

3.2.2.5 Diagramas de Casos de Uso

Um diagrama de casos de uso mostra um conjunto de casos de uso, atores e seus relacionamentos. Sua principal função é de apresentar um resumo do que o sistema deve fazer, bem como estabelecer os limites desse sistema (escopo).

A delimitação do sistema é representada por um retângulo rotulado com o nome do sistema. Dentro do retângulo, devem ficar os casos de uso (representando tudo o que o sistema deve fazer). Os atores, por não fazerem parte do sistema, devem ser colocados do lado de fora do retângulo. O diagrama de casos de uso também deve demonstrar os relacionamentos entre atores e casos de uso vistos anteriormente (figura 3.11).

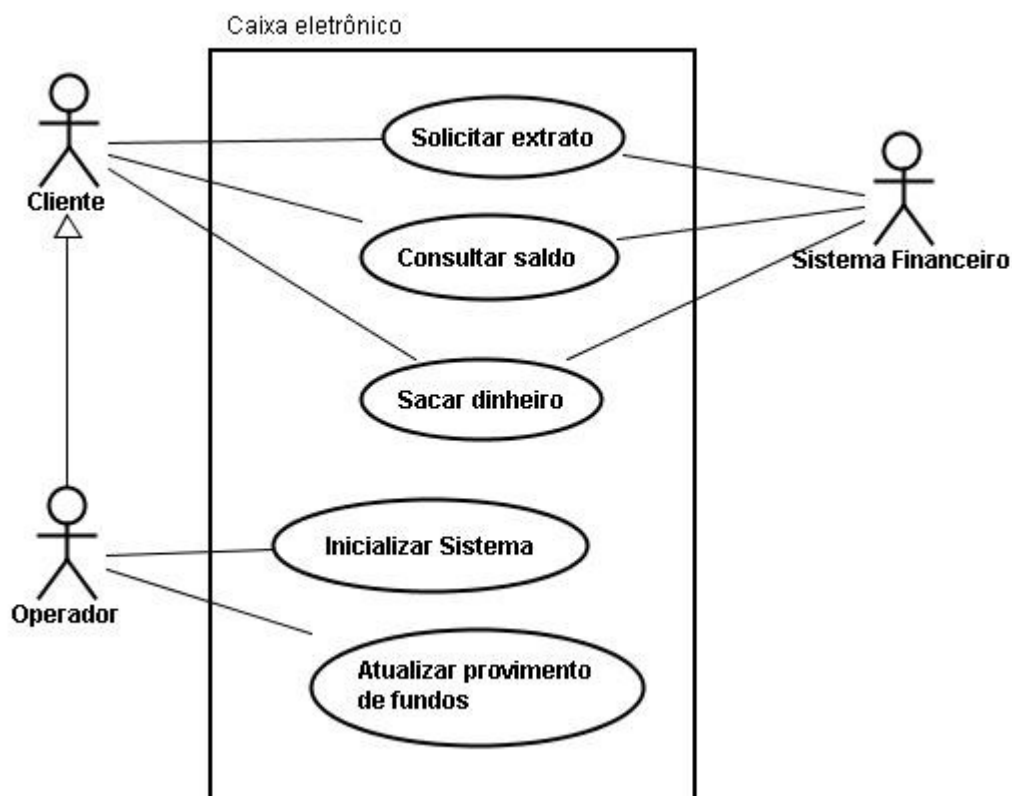


Figura 3.11: Diagrama de Casos de uso.

Eventualmente, um diagrama de casos de uso também poderá conter *pacotes*¹ que representem elementos do modelo de casos de uso. Com isso, sistemas muito complexos, podem ser subdivididos formando grupos de casos de uso de forma que possam ser compreendidos e gerenciados em partes.

3.2.2.6 Metamodelo de caso de uso

Os elementos do caso de uso podem ser representados por um metamodelo na UML através de um diagrama de classes. A figura 3.12 ilustra parte do metamodelo de caso de uso~~metamodelo apresentado na figura 3.12 é um subconjunto do metamodelo de casos~~ de uso especificado na UML 1.4 [OMG 2003].

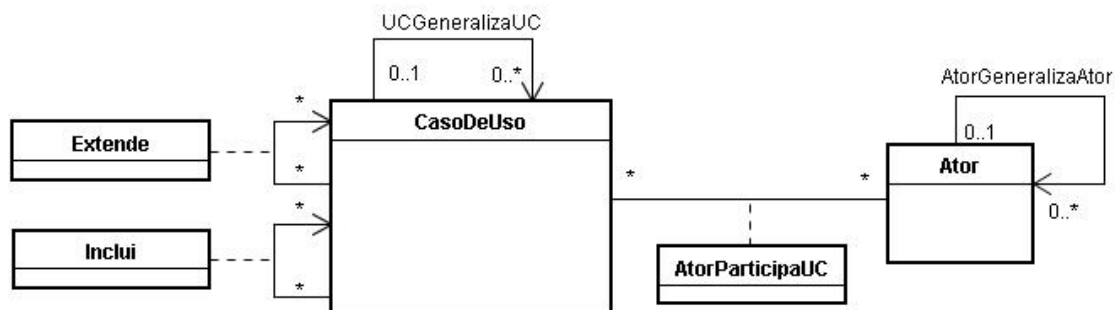


Figura 3.12: Metamodelo de caso de uso.

¹ Um pacote é um mecanismo de agrupamento definido na UML.

3.2.3 Estrutura básica de descrição

Basicamente, um caso de uso contém um nome e uma descrição. O nome do caso de uso deve ser único em relação aos demais casos de uso do sistema e, geralmente, representa o objetivo do caso de uso. A descrição contém os elementos básicos a seguir:

Atores: descrição dos atores envolvidos no caso de uso;

Pré-condições: regras de estado que definem como o sistema deve se encontrar antes de disparar o caso de uso;

Fluxo de eventos: atividades que ocorrem entre os atores e o sistema à medida que interagem para atingir um objetivo;

Pós-condições: regras de estado que definem como o sistema deve se encontrar depois de executado o caso de uso.

3.3 Processo de desenvolvimento dirigido a casos de uso

Um ciclo de vida de desenvolvimento iterativo e incremental se baseia no aumento e no refinamento sucessivo de um sistema através de múltiplos ciclos de desenvolvimento (iterações) [Larman 2004]. Cada um dos ciclos considera um subconjunto de requisitos. No próximo ciclo, um outro subconjunto de requisitos é considerado para ser desenvolvido, o que produz um novo incremento do sistema que contém extensões e refinamentos sobre o incremento anterior [Bezerra 2002].

O sistema cresce ao longo do tempo, pelo acréscimo de novas funcionalidades a cada iteração até que o sistema completo esteja construído (figura 3.13).

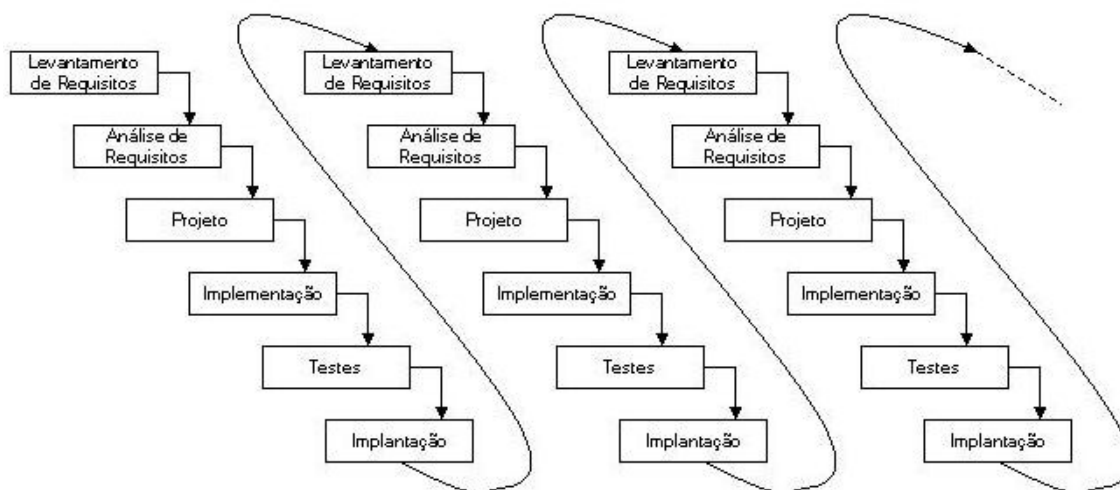


Figura 3.13: O modelo de desenvolvimento iterativo e incremental.

Fonte: Bezerra 2002

Na prática, cada iteração corresponde a um mini ciclo em cascata, sendo que não é necessário concluir todo um ciclo para se iniciar outro. Por exemplo: com o término da atividade de análise do ciclo 1, pode-se iniciar a análise do ciclo 2 em paralelo às fases de projeto e construção do ciclo 1.

Normalmente, cada ciclo de desenvolvimento deve ser planejado para ser executado de 2 semanas até 2 meses, sendo que podem ocorrer ciclos de desenvolvimento sem que haja entrega para o cliente.

Com base no modelo de desenvolvimento iterativo e incremental, Ivar Jacobson propôs em [Jacobson et al. 1992] que os ciclos de desenvolvimento sejam organizados por casos de uso, originando o termo **desenvolvimento dirigido a casos de uso** (*use case driven development*).

A cada ciclo é atribuído um subconjunto de casos de uso ou de versões simplificadas de casos de uso (o que é bastante comum quando o caso de uso completo é muito complexo para ser tratado em um ciclo) para ser desenvolvido. Geralmente, os casos de uso são agrupados em função da prioridade (importância do caso de uso para o cliente) e do risco de desenvolvimento. Uma boa estratégia é atacar os casos de uso de mais alta prioridade e criticidade nos ciclos de desenvolvimento iniciais.

Nesta abordagem, os casos de uso não são usados apenas para capturar requisitos funcionais e definir ciclos de desenvolvimento. Além de facilitar a identificação, organização e documentação dos requisitos funcionais, os casos de uso podem desempenhar um papel mais central e significativo no processo de desenvolvimento de software.

De fato, muitos processos de desenvolvimento de software, tal como o PU (Processo Unificado) [Kruchten 2000], se auto intitulam como sendo "dirigidos a casos de uso". Isto significa que os casos de uso são a base para todo o processo de desenvolvimento, pois dirigem atividades como as de análise, projeto, construção e testes.

É esta habilidade dos casos de uso em unificar as atividades de desenvolvimento que os torna um poderoso instrumento para planejamento e acompanhamento de projetos de desenvolvimento de software. Estes relacionamentos são diretamente refletidos nos demais modelos do processo de desenvolvimento conforme ilustrado na figura 3.14.

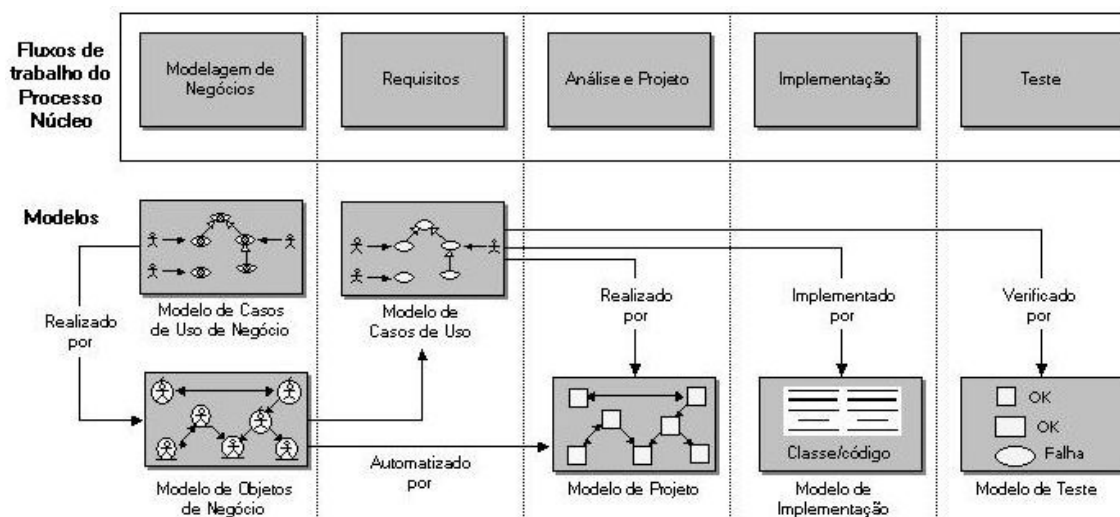


Figura 3.14: O modelo de casos de uso e seus relacionamentos com outros modelos de desenvolvimento.

Fonte: Kruchten 2000

Casos de uso também, também, podem ajudar a estruturar informações de gerenciamento do projeto e exercem especial influência nas especificações de projeto de interface com o usuário (IU), de testes do sistema e de documentação do usuário. Embora estas especificações não estejam nos casos de uso, elas se conectam a eles.

Cockburn usa a metáfora da roda para ilustrar esses conceitos. Na roda, os casos de uso representam o eixo e as outras informações são os raios levando a diferentes direções, conforme demonstra a figura 3.15.

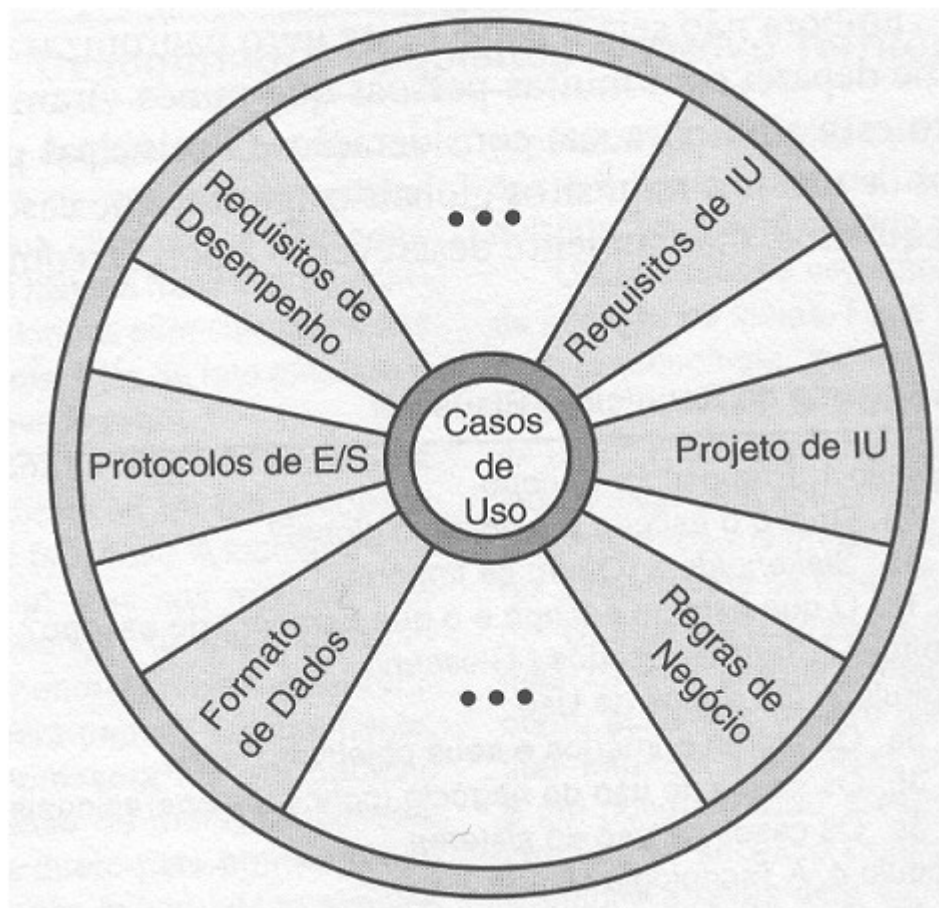


Figura 3.15: O modelo de requisitos “eixo-e-raios”.

Fonte: Cockburn 2005

3.4 Abordagens para reuso de casos de uso

A seguir, são apresentados os principais métodos e técnicas que, atualmente, podem contribuir para o reuso de casos de us~~interfaces com o usuário~~. Cada proposta será apresentada separadamente e será feita uma análise da sua aplicabilidade nas atividades de projeto de IU em um processo de desenvolvimento OO.

3.4.1 Casos de uso reusáveis

Considerando que os casos de uso servem como base para o projeto de IU, é razoável considerar o *reuso de casos de uso* como uma estratégia de reuso de interfaces. Para isso, é preciso definir o que torna um caso de uso reusável.

Biddle, Noble e Tempero salientam que a probabilidade de qualquer coisa ser reusada está relacionada com a forma com a qual ela é escrita. Eles observaram que os casos de uso escritos na forma essencial são aplicáveis em mais situações do que as formas convencionais de narrativa (casos de uso concretos), o que sugere que casos de uso essenciais são mais reusáveis [Biddle et al. 2002].

Além disso, com um caso de uso essencial é possível se obter vários casos de uso concretos em função de diferentes requisitos de IU, tais como diferentes plataformas de software ou de hardware. Desta forma, os casos de uso essenciais facilitam o reuso por quê permitem que muitos casos de uso concretos sejam especificados com base em apenas um caso de uso essencial.

O modelo de casos de uso prevê que um caso de uso possa referenciar a outros casos de uso através dos mecanismos de extensão, inclusão e generalização. A utilização efetiva destes mecanismos na modelagem de casos de uso contribui para o reuso, pois seqüências recorrentes de interação podem ser agrupadas em um caso de uso e serem reutilizadas por outros casos de uso. Desta forma, evita-se a redundância de interações.

Existem várias técnicas propostas de modelagem dos relacionamentos de casos de uso. Saeki, por exemplo, propôs uma técnica para extrair partes comuns ou similares das descrições dos casos de uso e colocá-las em casos de uso em separado que são relacionados pelos mecanismos de extensão, inclusão e generalização. Sua abordagem também permite capturar padrões na evolução dos casos de uso de acordo com as mudanças de requisitos. Para atingir isso, as partes dos casos de uso modificáveis (“*hotspots*”) e as não-modificáveis (“*frozen spots*”) são identificadas baseadas na avaliação das dependências de controle ou das dependências de dados encontradas entre os casos de uso [Saeki 1999, Saeki 2000].

Existe um forte debate sobre o uso e a semântica dos casos de uso [Cockburn & Fowler 1998, Lif 1999, Pimenta 2000, Constantine & Lockwood 2001, Jacobson 2003]. Pessoas diferentes estão usando casos de uso de forma distinta. Casos de uso são usados tanto para orientar o projeto OO quanto para a modelagem de tarefas e o projeto de interfaces.

No enfoque da engenharia de software, os casos de uso orientam o projeto OO para definir os objetos e como eles colaboram entre si para a realização dos casos de uso. Entretanto, quando os casos de uso estão nas mãos dos~~O projeto OO define os objetos e como eles devem colaborar entre si para realizar os casos de uso. Este é o enfoque da engenharia de software. Entretanto, enquanto os casos de uso estão na mão de~~ projetistas de interface, eles servem de modelo de tarefas para entendimento das necessidades do usuário e como um guia para o projeto de IU. Este é o enfoque da engenharia de usabilidade.

Modelar casos de uso de forma a contemplar estes dois enfoques ainda não é uma tarefa trivial, pois o ato de decompor casos de uso em vários casos de uso relacionados (como se fosse num processo de normalização de eventos) pode ser útil do ponto de vista da engenharia de software, mas também pode fazer com que o caso de uso perca a sua principal característica que é a simplicidade da narrativa. Desta forma, a legibilidade do caso de uso pelos usuários pode ficar comprometida.

Embora a UML implemente algum esforço de formalização de um caso de uso, ainda há muito que avançar neste tópico. Sem uma definição mais formal do fluxo de eventos do caso de uso, a utilidade dos relacionamentos entre casos de uso - como forma de aumentar o reuso - cai drasticamente.

Em [Rui & Butler 2003] é proposto um metamodelo de casos de uso que dá um maior grau de formalização ao modelo e permite a refatoração dos casos de uso. Entretanto, ainda não existe um consenso sobre qual meta-modelo é mais adequado para especificação de casos de uso.

Enquanto isso não ocorre, esse é um dos motivos pelo qual a técnica de decomposição de casos de uso ainda não rende os dividendos esperados de reuso nos projetos de software.

3.4.2 Casos de uso parametrizados

A reusabilidade de um caso de uso é afetada significativamente pelas dependências contidas nele. Um caso de uso terá reusabilidade limitada se ele contiver referências específicas ao sistema ou ao domínio da aplicação para o qual ele foi criado. Portanto, para compartilhar um caso de uso, é preciso endereçar estas dependências de forma que o mesmo caso de uso possa ser usado em situações diferentes [Biddle et al. 2002].

Uma das maneiras de reduzir estas dependências é através de customização. Customização é a habilidade de trocar partes de uma unidade sempre quando ela é chamada. Em linguagens de programação se usa a passagem de parâmetros, onde parâmetros são usados como variáveis que “guardam o lugar” em uma unidade de código. Essas variáveis são substituídas pelos valores que são passados para os parâmetros quando a unidade de código é chamada.

Uma abordagem apropriada de customização de caso de uso é a parametrização do texto da narrativa do caso de uso [Biddle et al. 2002, Cockburn 2005]. Este processo é semelhante ao que ocorre com o processamento de macros em um processador de texto. Parâmetros são embutidos em qualquer parte do texto para que sejam posteriormente substituídos por valores específicos. De posse destes valores, o processador da macro gera um texto novo, substituindo os parâmetros pelos valores no texto original.

Com isso, é possível descrever um caso de uso com parâmetros embutidos no corpo da narrativa, de forma a eliminar dependências. Desta forma, amplia-se a reusabilidade do caso de uso pois pode ser gerado para um sistema em particular a partir da especificação dos valores dos parâmetros.

Um exemplo de caso de uso parametrizado é demonstrado na tabela 3.3. A tabela 3.4 apresenta o mesmo caso de uso após a substituição dos parâmetros.

Tabela 3.3: Exemplo de caso de uso parametrizado

Listar #{entidade}s

Ator	Sistema
	Lista todas as #{entidade}s em ordem alfabética

Tabela 3.4: Exemplo de caso de uso após substituição de parâmetros

Listar contas

Ator	Sistema
	Lista todas as contas em ordem alfabética

3.4.3 Padrões de casos de uso (*use case patterns*)

Ao se projetar software usando casos de uso essenciais, observa-se que existem seqüências de interação no fluxo de eventos que recorrem dentro do mesmo sistema e, mais importante ainda, em outros sistemas. Estes sistemas nem precisam ser do mesmo domínio de negócio, pois podem ser tão diferentes quanto um sistema de controle de estoques e um sistema de controle de rede.

Ao investigar estas seqüências recorrentes no corpo dos casos de uso, constata-se que elas representam soluções para problemas particulares na modelagem de casos de uso: as seqüências são em comum porque elas fornecem soluções para problemas em comum ao modelar as interfaces dos sistemas com os atores que os rodeiam [Biddle et al. 2001].

Desta forma, é possível identificar *padrões de casos de uso (use case patterns)* [Saeki 1999, Biddle et al. 2001, Overgaard & Palmkvist 2005] que podem ser documentados e catalogados de forma semelhante a *padrões de projeto* [Gamma et al. 2000].

Entretanto, Biddle, Noble e Tempero [Biddle et al. 2001] salientam que os padrões de caso de uso estão filosoficamente mais próximos de *padrões de análise* [Fowler 1997] do que de *padrões de projeto*, pois da mesma forma que os padrões de análise, esses padrões também descrevem artefatos de análise ao invés de projetos OO. Mas há uma diferença básica entre padrões de casos de uso e padrões de análise: padrões de casos de uso essenciais descrevem padrões em casos de uso essenciais (diálogos característicos de intenções de usuário e responsabilidades do sistema) enquanto que os padrões de análise descrevem padrões em modelos no negócio de domínio.

Um padrão de caso de uso é aquele que pode ser reusado em função de representar uma tarefa (orientado a tarefa) ou um domínio (orientado a domínio) recorrente a vários sistemas.

Procura-se, sempre que possível, definir padrões de casos de uso parametrizados, ou seja, a parametrização do texto da narrativa do caso de uso de forma a aumentar as chances de reuso. Parâmetros tornam localizadas as referências específicas ao sistema ou ao domínio da aplicação para o qual ele foi criado e possibilita que o mesmo caso de uso possa ser usado em situações diferentes.

De uma forma geral, os padrões de casos de uso orientados a tarefas são mais facilmente parametrizáveis e reusáveis do que padrões de casos de uso orientados a domínios.

Para transformar as descrições de seqüências de interação em um padrão de caso de uso é preciso analisar as *forças* agindo no caso de uso, ou seja, as considerações importantes que estão impactando no caso de uso [Alexander et al. 1977]. Para os padrões de caso de uso essenciais, as forças capturam características da interação entre o ator e o sistema: questões de iniciativa, fluxo de informação e usabilidade. Cada padrão tem conseqüências positivas e negativas em algumas destas forças.

Um exemplo de padrão de caso de uso é apresentado na figura 3.16, o “Caso de uso de Alarme”. O problema que este padrão resolve é escrever um caso de uso essencial que modele uma interação onde o sistema precisa notificar um ator sobre um evento importante, tal como uma mudança no seu estado interno ou uma potencial violação de uma regra de negócio.

Caso de uso de alarme

Como você faz o sistema informar o usuário sobre algo?

Forças

- O sistema precisa chamar a atenção do usuário para uma mudança no seu estado interno.
- O sistema está prestes a quebrar uma regra de negócio.
- A notificação deve ser assíncrona, ou seja, os atores não devem ter que disparar o caso de uso.

Portanto: Escreva um caso de uso que comece com o sistema tendo a responsabilidade de avisar o usuário.

Exemplo

Avisa início de Apresentação

Intenção do usuário	Responsabilidade do sistema
	emite sinal de “apresentação vai começar” mostra o nome, teatro e tempo da apresentação

Conseqüências

- + O sistema assume a responsabilidade por iniciar o caso de uso.
- + O sistema pode passar informação sobre o alarme para o ator.
- + O ator não tem que interromper imediatamente a tarefa que estiver fazendo para responder ao alarme.
- O ator pode ignorar o alarme.

Variante

Se o alarme é importante, você pode precisar incluir um passo de confirmação:

Avisa lotação esgotada

Intenção do usuário	Responsabilidade do sistema
	emite sinal de “lotação esgotada” mostra o nome, teatro e data e hora da apresentação
confirma o aviso	

Esta variante tem as seguintes consequências diferentes do padrão principal:

- O ator não pode continuar com a tarefa que está fazendo: ele deve interrompê-la para conformar o alarme.
- + O ator não pode ignorar o alarme

Figura 3.16: Exemplo de padrão de caso de uso.

Fonte: Biddle et al. 2001

Um catálogo de padrões de caso de uso é muito útil para construir facilmente descrições de caso de uso~~descrições de caso de uso facilmente~~. Os padrões dependem de domínios de problema, daí que é necessária uma técnica que mostre como extrair padrões de caso de uso como componentes reusáveis das reais descrições de caso de uso, e como armazená-las em um tipo de sistema de base de dados, chamado de base de padrões [Saeki 1999]

Um dos problemas com os padrões de casos de uso é o de encontrar um padrão de caso de uso que corresponda à funcionalidade sendo modelada~~caso de uso que atenda~~ (catalogação e pesquisa). Vários autores propõem mecanismos para minimizar este problema. Woo e Robinson [Woo & Robinson 2002], por exemplo, propõem que, a partir de uma especificação da narrativa de um caso de uso em um diagrama da UML (diagrama de atividades, por exemplo), seja possível submeter este diagrama como se fosse uma “query” e recuperar todos os casos de uso que contenham uma seqüência de interação o mais próxima possível da narrativa original. Isto não permite que a narrativa seja reusada (já que foi preciso descrevê-la *antes* de submeter a “query”) mas pode levar ao reuso da realização do caso de uso a partir das especificações do caso de uso padrão encontrado.

3.4.4 Padrões de domínio

Os padrões de casos de uso são muito mais úteis se puderem ser relacionados uns aos outros e/ou agrupados por tarefas ou domínios em comum. Constantine e Lockwood propõem que, além dos relacionamentos de inclusão, extensão e generalização, os casos de uso sejam relacionados por afinidade (semelhança e equivalência) [Constantine & Lockwood 1999]. Além destes relacionamentos, os casos de uso que participam na solução de um determinado problema podem compor uma linguagem de padrões de casos de uso a qual chamamos de padrão de domínio.

Um padrão de domínio descreve o conjunto de casos de uso que são re-correntes na modelagem de casos de uso de sistemas de um mesmo domínio de problema e que se relacionam entre si e colaboram para dar suporte à prestação de uma funcionalidade de valor para os usuários. Um padrão de domínio de comércio eletrônico, por exemplo, pode conter os casos de uso: “Consulta um produto”, “Busca por palavra-chave”, “Busca por categoria”, “Coloca no carrinho de compras”, “Efetua compra”, entre outros, pois estes casos de uso são comuns a todos os sistemas de comércio eletrônico. Da mesma forma, o conjunto de casos de uso que suportam o serviço de autenticação e de permissões de usuários, e o conjunto de casos de uso que suportam o serviço de publicação de notícias em um site na Web, são exemplos de padrões de domínio.

4 PROJETO E REUSO DE INTERFACES COM O USUÁRIO

Neste capítulo, é feita uma discussão dos papéis envolvidos nas atividades de projeto de IU através do enfoque da engenharia de software e da engenharia de usabilidade, e como estas atividades são desempenhadas em um processo de desenvolvimento OO dirigido por casos de uso. Os principais instrumentos utilizados para a modelagem das interfaces são apresentados e analisados na seção 4.2. Finalmente, na seção 4.3o processo de desenvolvimento de um software através do enfoque da engenharia de software e da engenharia de usabilidade e como estas atividades são desempenhadas em um processo de desenvolvimento OO dirigido por casos de uso. Os principais instrumentos utilizados para a modelagem das interfaces são apresentados e, finalmente, é feita uma revisão das principais abordagens de reuso de interface utilizadas atualmente.

4.1 Projeto de interface com o usuário no processo de desenvolvimento

O desenvolvimento de sistemas interativos é um processo eminentemente multidisciplinar requerendo idealmente o trabalho de uma equipe formada por profissionais das várias áreas envolvidas (como p.ex. Design Visual e Gráfico, Ergonomia e Engenharia de Software entre outras). Na prática do desenvolvimento, contudo, é reconhecida a enorme dificuldade de comunicação e integração entre estes membros de equipes multidisciplinares, devido provavelmente à dificuldade de integração de conceitos de suas disciplinas: cada uma possui seu vocabulário específico, suas expressões e notações, suas formas de organizar o desenvolvimento, etc. Inúmeros trabalhos de pesquisa [ICSE 2003, Seffah et al. 2005] convergem para a idéia central de que para desenvolver sistemas interativos que sejam úteis e usáveis tem-se necessidade de uma integração sistemática e correspondências explícitas entre a variedade de teorias, modelos, técnicas e ferramentas das diferentes áreas e que possibilite concretamente um desenvolvimento em equipe mais efetivo. Nosso trabalho nos últimos anos tem sido um esforço interdisciplinar de integração das áreas de Engenharia de Software (abreviada ES) e Interação Humano-Computadorrecentes de pesquisa [ICSE 2003, Seffah et al. 2005] convergem para a idéia central de que para desenvolver sistemas interativos que sejam úteis e usáveis tem-se necessidade de uma integração sistemática e correspondências explícitas entre a variedade de teorias, modelos, técnicas e ferramentas das diferentes áreas e que possibilite concretamente um desenvolvimento em equipe mais efetivo. Nosso trabalho nos últimos anos tem sido um esforço

~~interdisciplinar de integração das áreas de Engenharia de Software (abreviada ES) e Interação Homem-Sistemas Computacionais (abreviada IHC) e segue uma tendência atual de mudança de atitude da ES em face de fatores humanos.~~

A interseção entre as áreas de ES e IHC não é ainda nem natural nem objetiva. Entretanto, as comunidades de ES e IHC são unânimes em reconhecer que um sistema interativo só é bem sucedido se atende às expectativas de seus usuários [Denning 1994, Cox 1993]. No âmbito da ES, o termo "expectativas" tem sido tipicamente entendido como sinônimo de "requisitos funcionais", dando pouca ou nenhuma atenção aos requisitos não funcionais de usabilidade ou de interação. Claramente, enfoques tradicionais da ES e mesmo os enfoques orientados a objeto mais atuais (p.ex. Processo Unificado (PU) [Kruchten 2000], e outros baseados na notação UML) mostram-se frequentemente inoperantes e inadequados para desenvolver sistemas interativos, não contemplando o ponto de vista do usuário e não prevendo explicitamente o projeto das interfaces com o usuário (abreviadas IU) como uma de suas etapas. Esta lacuna está na origem de um problema de integração multidisciplinar que se manifesta desde o início do planejamento do sistema e continua durante todo o ciclo de vida do sistema interativo [Barthet 1988]: cada área considera aspectos diferentes e muitas vezes disjuntos do sistema sem nenhuma correspondência explícita e sistematicamente estabelecida entre eles. Sem esta correspondência, não há interação nem integração possível entre os especialistas de cada área (projetista de software e designer de interface). Entretanto, o projeto de IUs é uma atividade importante que necessita muito esforço da equipe e é considerada como um fator crítico de sucesso nos projetos de software da atualidade [Procaccino et al. 2005]. Portanto, devem ~~U e outros baseados na notação UML)~~ mostram-se frequentemente inoperantes e inadequados para desenvolver sistemas interativos não contemplando o ponto de vista do usuário e não prevendo explicitamente o projeto das interfaces com o usuário (abreviadas IU) como uma de suas etapas. Esta lacuna está na origem de um problema de integração multidisciplinar que se manifesta desde o início do planejamento do sistema e continua durante todo o ciclo de vida do sistema interativo [Barthet 1988]: cada área considera aspectos diferentes e muitas vezes disjuntos do sistema sem nenhuma correspondência explícita e sistematicamente estabelecida entre eles. Sem esta correspondência, não há interação nem integração possível entre os especialistas de cada área (projetista de software e designer de interface). Entretanto, o projeto de IUs é uma atividade importante que necessita muito esforço da equipe e é considerada como um fator crítico de sucesso nos projetos de software da atualidade [Procaccino et al. 2005]. Portanto, deve-se propor estratégias em que esta atividade esteja mais claramente descrita no processo de desenvolvimento.

Tendo como referência o Processo Unificado (PU), o projeto de IU ocorre após a definição dos requisitos funcionais do sistema que, na UML, são representados pelo modelo de casos de uso. ~~[Kruchten 2000], o projeto de interfaces com o usuário ocorre após a definição dos requisitos funcionais do sistema que, na UML, são representados nos diagramas de casos de uso.~~

O papel dos casos de uso na especificação de interfaces com o usuário ainda é uma questão em aberto. Existem ainda muitas dificuldades e carências de métodos e técnicas para se obter interfaces com qualidade/usabilidade a partir de uma especificação de caso de uso. Em particular, um dos problemas é de que a engenharia de requisitos tem usualmente priorizado sua atenção para os requisitos funcionais dando pouca ou nenhuma atenção aos requisitos de usabilidade ou de interação. Assim, mesmo enfoques

orientados a objetos mais atuais mostram-se freqüentemente inoperantes e inadequados para desenvolver sistemas interativos: a principal reclamação que se pode fazer a estes enfoques tradicionais é que eles não incitam os analistas a complementar sua visão orientada ao sistema com o ponto de vista do usuário. Conseqüentemente eles são levados a deduzir a concepção da interface a partir da *lógica de funcionamento* das funções da aplicação e não a partir da *lógica de utilização* do sistema como um suporte à realização de tarefas dos usuários [Pimenta 2000].

Um caso de uso passa por fases (e correspondentes níveis de abstração e de refinamento de descrição) ao longo do seu ciclo de vida: inicialmente, a descrição de um caso de uso consiste de um enunciado sucinto de seu objetivo. A partir daí, faz-se o detalhamento desta descrição do caso de uso com a definição dos passos do fluxo principal e fluxos alternativos, bem como os possíveis relacionamentos (inclusão, extensão e generalização) com outros casos de uso. Estas são algumas das atividades que constituem a modelagem de casos de uso [Bittner & Spence 2003].

Em um processo de desenvolvimento dirigido por casos de uso, é a partir das descrições dos casos de uso que se desenvolve o projeto da IU. Atualmente, é senso comum ~~de~~ que estas descrições devem estar no grau de abstração essencial [Constantine & Lockwood 1999, Cockburn 2005]. Na narrativa essencial, são explicitadas as interações entre o ator e o sistema sem descrever detalhes de como e onde estas interações serão realizadas.

A partir da descrição dos casos de uso essenciais, constrói-se um *protótipo* da interface com o usuário. O objetivo do protótipo é ajudar a validar os requisitos funcionais especificados nos casos de uso e validar se a interface está implementando adequadamente as interações necessárias para que o caso de uso atinja o objetivo.

Para construir um protótipo, o ambiente de interação deve ser definido, isto é, devem ser identificados os elementos de interação que vão ser utilizados (a tecnologia da interface) e o contexto no qual se dará a interação. Entretanto, a tecnologia de implementação destas interfaces não precisa necessariamente ter sido definida. Pode-se por exemplo fazer o projeto de interface de uma aplicação gráfica desktop a ser executada no ambiente Gnome/Linux sem saber qual a linguagem de programação e a arquitetura da aplicação que serão utilizadas para a sua construção.

O protótipo é, então, apresentado para um ou mais usuários que podem simular a execução das tarefas que serão suportadas pelo sistema e, a partir daí, fazer críticas acerca de uma ou outra característica. Além do feedback dos usuários, a equipe do projeto pode observar os usuários em ação e, com isso, identificar eventuais dificuldades na realização das tarefas, no uso das interfaces, bem como identificar discrepâncias nos requisitos funcionais. O protótipo é então corrigido ou refinado de acordo com tais críticas. Esse processo de revisão e refinamento continua iterativamente até que o protótipo atinja um nível satisfatório e seja aceito pelos usuários.

Os protótipos e as especificações das IUs constituem um conjunto de artefatos que chamamos de *casos de uso apresentados*. Os casos de uso apresentados são a forma visual e perceptível da descrição dos fluxos dos casos de uso essenciais. Eles ainda não fazem referência aos objetos de negócio do sistema, pois representam um nível intermediário entre a narrativa conceitual definida no modelo de casos de uso e os objetos e suas colaborações definidos na fase de realização dos casos de uso.

A fase que agrupa o conjunto das atividades necessárias para a obtenção dos casos de uso apresentados é chamada de *apresentação de caso de uso* (*use case presentation*).

Com os casos de uso apresentados definidos e validados, passa-se para a atividade de projeto OO, onde são traduzidos os elementos da IU em termos de objetos e classes e é especificado como esses objetos e classes colaboram entre si para a realização do caso de uso. Esta fase é chamada de *realização do caso de uso* (*use case realization*) [Jacobson et al. 1992].-

~~O papel dos casos de uso na especificação de interfaces com o usuário ainda é uma questão em aberto. Existem ainda muitas dificuldades e carências de métodos e técnicas para se obter interfaces com qualidade/usabilidade a partir de uma especificação de caso de uso. Em particular, um dos problemas é de que a engenharia de requisitos tem usualmente priorizado sua atenção para os requisitos funcionais dando pouca ou nenhuma atenção aos requisitos de usabilidade ou de interação. Assim, mesmo enfoques orientados a objetos mais atuais mostram-se frequentemente inoperantes e inadequados para desenvolver sistemas interativos: a principal reclamação que se pode fazer a estes enfoques tradicionais é que eles não incitam os analistas a complementar sua visão orientada ao sistema com o ponto de vista do usuário. Conseqüentemente eles são levados a deduzir a concepção da interface a partir da lógica de funcionamento das funções da aplicação e não a partir da lógica de utilização do sistema como um suporte à realização de tarefas dos usuários [Pimenta 2000].~~

4.2 Modelos de especificação de interface com o usuário

Nesta seção revisamos algumas abordagens para a modelagem de IUs. Estas abordagens vão desde especificações informais e de formato livre, tais como as encontradas nos protótipos (seção 4.2.1) até especificações formais implementadas por linguagens de especificação de IUs (seção 4.2.4).

4.2.1 Protótipos

Protótipos, atualmente, são considerados como um dos instrumentos mais eficazes para a modelagem de IUs.

Um protótipo é uma representação limitada (um croqui ou esboço) do projeto da interface que permite aos usuários ter uma visão concreta e visual da interface do sistema e explorar a sua conveniência. Ele possibilita que os usuários interajam com um produto imaginado visando a adquirir alguma experiência de como utilizá-lo em um ambiente real e a explorar os usos para ele imaginados [Preece et al. 2005].

Protótipos podem ser tão simples quanto um esboço de uma tela ou conjunto de telas rabiscadas em papel ou tão complexos quanto um conjunto de telas eletrônicas que já se assemelham em muito com o software a ser construído. Na verdade, um protótipo de IU pode ser qualquer coisa, desde um *storyboard* de papel a uma parte complexa de um software.

Essa diversidade de protótipos pode ser classificada em duas categorias: protótipos de baixa-fidelidade e protótipos de alta-fidelidade. Cada uma destas categorias contém características próprias, vantagens e desvantagens que serão discutidas a seguir.

4.2.1.1 Protótipos de baixa-fidelidade.

Um protótipo de baixa fidelidade é aquele que não se assemelha muito ao produto final. Estes protótipos utilizam materiais muito diferentes da versão final pretendida, tais como papel e cartolina ou simples desenhos digitalizados.

Basicamente, um protótipo de baixa-fidelidade é um esboço (*sketch*) de uma tela, de um conjunto de telas ou de um *storyboard* com as seqüências de interações. Estes esboços podem ser construídos em papel (*paper sketches*) ou desenhados em computador.

Fazer protótipos em papel é rápido, fácil e dá grande flexibilidade para organizar os elementos da interface; dá pra refazer, validar, jogar fora e começar tudo de novo. Tudo isso a um custo mínimo e sem grandes desgastes [\[Preece et al. 2005\]](#).

Os protótipos em papel permitem não só esboçar a IU, mas também trabalhar com a interação. É comum haver sessões com os usuários de simulações de interação mediante a apresentação da seqüência de interfaces que serão utilizadas por eles para a realização de uma tarefa.

Protótipos desenhados por computador podem ser feitos até com ferramentas simples de software de desenho, tais como os programas de apresentação Microsoft Powerpoint e OpenOffice.org Impress ou os programas de desenho Corel Draw e Gimp.

Fazer protótipos desenhados por computador - apesar de exigir mais recursos que protótipos em papel - permite a prática de reuso de outros esboços, bem como a utilização de *templates* de “fotografias” de telas (*screenshots*) como base para o desenho do protótipo. Também, além de produzirem imagens mais acuradas do sistema, permitem introduzir alguma experiência de interação do usuário pelo encadeamento das telas através de *hyperlinks*.

As figuras 4.1 e 4.2 apresentam, respectivamente, um exemplo de protótipo em papel e um exemplo de protótipo desenhado por computador.

1

Nome do operador: Fulano de Tal e tal

NOVA NOTÍCIA

TÍTULO:

TEXTO:

SINOPSE:

DATA PUBLICAÇÃO:

AUTOR:

FONTE:

NOTÍCIAS

Figura 4.1: Exemplo de protótipo em papel.

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX

E-MAIL SENHA AJUDA SAÍDA

Módulo Administração

Nova – Notícia

Preencha os campos abaixo para criar a noticia (campos em negrito são obrigatórios):

Título da notícia:

Texto da notícia:

Link N I A

Sinopse da notícia:

Data para publicação:

Autor:

Fonte da Notícia:

Manchete no Site:

Nome do operador:
Fulano de Tal e tal

Noticias:

Objeto 2.

Objeto 2.

Objeto 2.

Objeto 2.

Nota: não existe o campo publicado nesse diálogo pq existe o botão publicar.

Figura 4.2: Exemplo de protótipo desenhado por computador.

Protótipos de baixa fidelidade são úteis porque tendem a ser simples, baratos e de rápida produção. Isso também significa que podem ser rapidamente modificados, oferecendo, portanto, suporte à exploração de design e de idéias alternativas. Tal vantagem é particularmente importante nos primeiros estágios de desenvolvimento, uma vez que os protótipos utilizados para explorar idéias devem ser flexíveis e encorajar - ao invés de desencorajar - a exploração e a modificação.

O projetista que constrói protótipos de baixa-fidelidade está focado na interação, nos componentes da IU, e na estrutura geral do sistema, deixando em plano secundário os aspectos de design como aparência, layout e estilos de design.

4.2.1.2 Protótipos de alta-fidelidade.

A prototipação de alta-fidelidade utiliza materiais que são esperados que estejam no produto final.

Um protótipo de alta-fidelidade é um software que apresenta as IUs e simula o seu funcionamento de forma bem próxima do que seria se o software estivesse pronto. Por serem programas de computador, os protótipos de alta-fidelidade também são conhecidos como protótipos executáveis [Preece et al. 2005].

Os protótipos executáveis permitem mostrar de forma mais fidedigna como o software vai ser quando estiver pronto. Com eles, é possível ajustar o layout definitivo e, principalmente, pode-se testar a interação com os reais dispositivos de interação (mouse, teclado, monitor de vídeo, etc).

Para construir um protótipo em software, obviamente será necessária uma ferramenta de software que ofereça suporte a essa atividade. Geralmente, esses protótipos são construídos com o auxílio de ferramentas de programação visual dos ambientes integrados de desenvolvimento (IDEs) de uma determinada tecnologia. Estas ferramentas permitem programar as IUs de forma gráfica com um mínimo de necessidade de escrever diretamente código de programação para obter um protótipo executável. Trata-se de ambientes de desenvolvimento completos e, portanto, de ferramentas poderosas. Ainda assim, pode ser bastante simples realizar protótipos com elas. São exemplos de IDEs: o *Adobe Dreamweaver* [Adobe 2007] para sistemas na Web e o *NetBeans IDE* [NetBeans 2007] e *Eclipse* [Eclipse 2007] ~~Macromedia-Dreamweaver para sistemas na Web e o NetBeans e Eclipse para sistemas GUI desktop.~~

Construir um protótipo executável não significa dar início à fase de construção do software, pois o projetista de IU, ao construir o protótipo, está focado na navegação e no uso do sistema e não na implementação de regras de negócio, de acesso e armazenamento das informações, e – principalmente – no estilo de programação e na arquitetura de software com que o código deve ser escrito. É comum que os programadores, ao receberem protótipos executáveis como artefato de especificação de IU, aproveitem muito pouco do código destes protótipos ou até mesmo não aproveitem nada. Protótipos executáveis, embora seja desejável, não necessariamente precisam ser escritos na mesma linguagem de programação a ser adotada na construção do software.

[Retfig 1994] argumenta que os projetos deveriam utilizar mais a prototipação de baixa-fidelidade devido aos problemas inerentes a prototipação de alta-fidelidade. Eis alguns problemas de projetos realizados com a prototipação de alta fidelidade:

- Levam muito tempo para ser construído;

- Os revisores e aplicadores de testes tendem a comentar aspectos superficiais, em vez do conteúdo;
- Os desenvolvedores relutam em mudar algo no qual trabalharam artesanalmente por horas;
- Um protótipo em software pode elevar demais as expectativas;
- É necessário apenas um *bug* em um protótipo de alta-fidelidade para interromper o teste.

[Rudd et al. 1996] apresentam mais algumas vantagens e desvantagens acerca dos dois tipos de prototipação conforme apresentado na tabela 4.1.

Tabela 4.1: Eficácia relativa de protótipos de baixa e de alta-fidelidade
Fonte: Rudd et al. 1996

Tipo	Vantagens	Desvantagens
Protótipo de baixa-fidelidade	<ul style="list-style-type: none"> ▪Custo mais baixo de desenvolvimento ▪Avalia múltiplos conceitos de design ▪Instrumento de comunicação útil ▪Aborda questões de layout de tela ▪Útil para identificação de requisitos de mercado ▪Prova-de-conceito (demonstrações de que o conceito funciona) 	<ul style="list-style-type: none"> ▪Verificação limitada de erros ▪Especificação pobre em detalhe para codificação ▪“Uso” conduzido pelo facilitador ▪Utilidade limitada após estabelecimento dos requisitos ▪Utilidade limitada para testes de usabilidade ▪Limitações de fluxo e navegação
Protótipo de alta-fidelidade	<ul style="list-style-type: none"> ▪Funcionalidade “completa” ▪Totalmente interativo ▪Uso conduzido pelo usuário ▪Define claramente o esquema de navegação ▪Uso para exploração e teste ▪Mesmo <i>look and feel</i> do produto final ▪Serve como uma especificação viva ▪Ferramenta de venda e marketing 	<ul style="list-style-type: none"> ▪Desenvolvimento mais caro ▪Sua criação demanda tempo ▪Ineficiente para designs prova-de-conceito (demonstrações de que o conceito funciona) ▪Não serve para coleta de requisitos

4.2.2 Modelos de Interface de Usuário

Os modelos de interface de usuário têm o propósito de descrever abstratamente a interface de usuário a ser criada, e podem ser definidos como uma descrição formal, declarativa e livre de implementação da interface [Eisenstein et al. 2000]. O modelo descreve em alto nível a implementação da interface, utilizando-se de componentes abstratos, os quais serão mapeados para componentes de interface da tecnologia alvo da aplicação.

Uma interface com o usuário pode ser totalmente descrita por um conjunto de objetos de interação (*interaction objects- IOs*). Um objeto de interação é qualquer elemento que permita, aos usuários de uma aplicação, visualizar ou manipular informações ou realizar uma tarefa interativa. Do ponto de vista da modelagem da IU, estes objetos são muitas vezes vistos como os blocos de construção atômicos da IU.

Vanderdonckt e Bodart [Vanderdonckt & Bodart 1993] identificam dois tipos de objeto de interação, o abstrato (*abstract interaction object - AIO*) e o concreto (*concrete interaction object - CIO*).

Um CIO é definido como qualquer entidade de IU que o usuário pode perceber (tais como texto, imagem e som) e/ou manipular (tais como botões, caixas de listagem ou menus). CIOs são objetos gráficos para captar e apresentar os dados relacionados à tarefa interativa do usuário. Um CIO também é conhecido como *widget* (*window gadget*) ou *controle*.

Um CIO contém duas características principais: a apresentação gráfica determinada pelo ambiente de interação (ex: MS-Windows XP, Gnome, OSF/Motif, ...) e o comportamento do objeto com as restrições (*constraints*) que devem ser respeitadas quando o usuário manipula o objeto.

Ao escolher um objeto, o projetista tem que prestar mais atenção no aspecto comportamental do objeto pois o poder de comunicação da IU e a sua facilidade de uso são pontos cruciais na escolha do IO [Vanderdonckt & Bodart 1993]. Neste caso, a definição da apresentação da IU passa a ser uma tarefa acessória a ser deixada para o artista gráfico. Esta observação convida a definir IOs independentemente de sua apresentação, mas não independentemente de seu comportamento.

O objeto de interação abstrato (AIO) apresenta uma visão abstrata do CIO onde as características físicas são independentes do ambiente de interação a ser utilizado. Por definição, um AIO não tem aparência gráfica, mas cada AIO é conectado a 0, 1 ou muitos CIOs em diferentes ambientes de interação com diferentes nomes e apresentações. AIOs são mapeados em CIOs para que se possa produzir uma interface observável. A figura 4.3 ilustra estas relações.

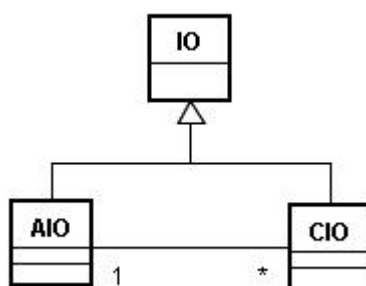


Figura 4.3: Objetos de interação abstratos e concretos

Cada AIO é identificado por um único nome genérico (por ex: janela, caixa de seleção), por atributos abstratos genéricos (por ex: altura, largura, cor) e por atributos abstratos genéricos (por ex: altura, largura, cor) e eventos abstratos (por ex: seleção de valor, clique de mouse).

Cada IO pode ser simples ou composto (contendo dois ou mais IOs simples). Um IO composto contém uma hierarquia de IOs simples e de IOs compostos que, combinados, colaboram e se referenciam entre si para compor um novo IO. Desta forma, um modelo

de IU nada mais é do que uma hierarquia de IOs onde a própria IU é um IO composto no mais alto nível da hierarquia.

Os IOs são compostos por classificados em 3 tipos distintos: os que representam os espaços de interação, os que representam os elementos primitivos de interação e os que representam os mecanismos de interação. A figura 4.4 apresenta estes tipos e as possíveis combinações entre eles.

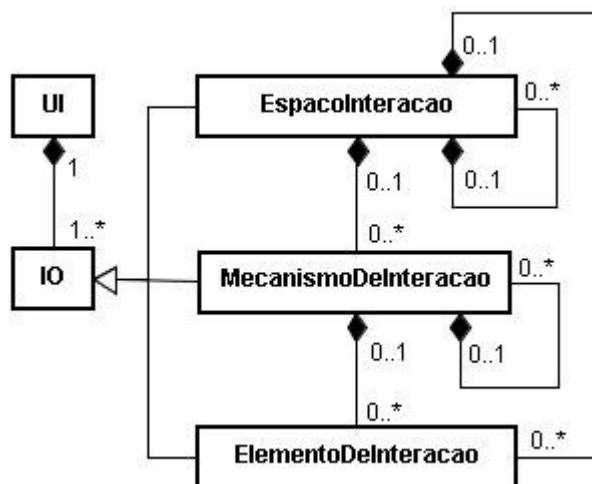


Figura 4.4: Tipos de objetos de interação

Na figura, uma IU deve ser constituída de pelo menos um IO e pode utilizar um ou mais espaços de interação onde devem acontecer as interações do usuário com o sistema. Cada espaço de interação contém um conjunto limitado e organizado de mecanismos de interação e de elementos de interação. Os elementos de interação são os menores objetos de interação com os quais uma pessoa pode interagir, logo, um elemento de interação é indivisível.

Os elementos de interação podem ser combinados de forma a implementar interações mais complexas. Este conjunto de elementos de interação, compõe um mecanismo de interação (ou seja, um IO composto). Como exemplo de mecanismo de interação, considere o conjunto dos elementos de interação de caixa de texto, botão e calendário. Quando estes elementos são organizados juntos na interface, implementam um mecanismo de interação para o preenchimento de uma data pelo usuário.

Examinando os diferentes ambientes de interação, constata-se que existem conjuntos limitados de IOs que compõem estes ambientes. O conjunto desses IOs formam o *vocabulário* deste ambiente [UIML 2004]. Desta forma, podem existir *vocabulários abstratos ou genéricos*, que são independentes de ambiente de interação (formados por AIOs), e *vocabulários concretos* (formados por CIOs de cada ambiente de interação).

Os vocabulários nos permitem agregar uma ontologia completa de AIOs e CIOs classificando-os pelas suas capacidades interativas.

4.2.3 Um modelo de referência para IUs

Como vimos na seção anterior, uma IU pode ser descrita através de hierarquias de objetos de interação e como eles se comportam. Estas hierarquias podem ser representadas em vários níveis de abstração que vão desde descrições conceituais até as implementações da IU em uma determinada tecnologia.

Uma representação destes níveis de abstração de IU é definida no modelo de desenvolvimento de IUs para aplicações interativas multi-contexto, o CRF - *Cameleon Reference Framework* [Calvary et al. 2003]. O *Cameleon Framework* define modelos que suportam os vários níveis de descrições de IU (que no CRF são chamados de passos de desenvolvimento) em vários contextos de uso.

A figura 4.5 contém uma versão simplificada do CRF que apresenta os seus 4 passos de desenvolvimento (Task & Concepts (T&C), Abstract UI (AUI), Concrete UI (CUI) e Final UI (FUI)), bem como os processos de transformação de artefatos entre os passos.

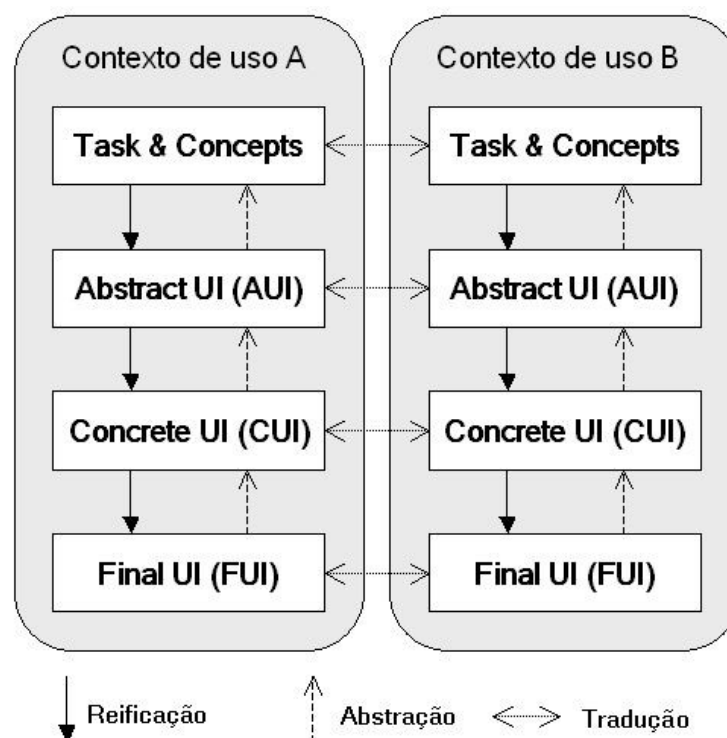


Figura 4.5: O Cameleon Reference Framework.

Final UI (FUI): é uma IU operacional, ou seja, qualquer IU executando em uma plataforma de computação específica. Os artefatos de uma FUI são os códigos-fonte de linguagens que implementam a IU, tais como Java e HTML, os códigos-objeto (executáveis) que renderizam a IU. Portanto, uma FUI pode ser interpretada, compilada e executada em uma determinada plataforma.

Final UI (FUI) é uma IU operacional, ou seja, qualquer IU executando em uma plataforma de computação específica. Os artefatos de uma FUI são os códigos-fonte de linguagens que implementam a IU, tais como Java e HTML, os códigos-objeto (executáveis) que renderizam a IU. Portanto, uma FUI pode ser interpretada, compilada e executada em uma determinada plataforma.

Concrete UI (CUI) é um modelo de IU que permite a especificação da aparência e do comportamento da IU através da descrição dos objetos de interação que podem ser percebidos pelos usuários. Portanto, uma CUI consiste de uma decomposição hierárquica de CIOs de um determinado vocabulário em um dado contexto de uso.

~~*Concrete UI (CUI): é um modelo de IU que permite a especificação da aparência e do comportamento da IU através da descrição dos objetos de interação que podem ser percebidos pelos usuários. Portanto, uma CUI consiste de uma decomposição hierárquica de CIOs de um determinado vocabulário em um dado contexto de uso.*~~

A CUI é uma representação abstrata de uma FUI de forma que seja independente de qualquer plataforma computacional ou das linguagens de programação usadas para desenvolver IUs. Entretanto, CUIs dependem de ambiente de interação, ou seja, uma instância de uma CUI endereça um único ambiente de interação por vez.

Embora a CUI possa explicitar o “*look & feel*” final de uma FUI, ela ainda é uma maquete da IU que executa somente dentro de ambientes de desenvolvimento de IUs.

Os artefatos de uma CUI, geralmente, são especificações em XML da hierarquia de CIOs que compõe a IU.

A Abstract UI (AUI) é considerada como uma abstração de uma CUI com respeito à modalidade. Uma AUI é uma expressão canônica da renderização dos conceitos de domínio e de tarefas de forma que seja independente dos objetos de interação disponíveis nos ambientes de interação.

~~*Abstract UI (AUI): Uma AUI é considerada como uma abstração de uma CUI com respeito à modalidade. Uma AUI é uma expressão canônica da renderização dos conceitos de domínio e de tarefas de forma que seja independente dos objetos de interação disponíveis nos ambientes de interação.*~~

A AUI define *containers* abstratos e componentes individuais através do agrupamento de tarefas de acordo com certos critérios (por ex: padrões estruturais de modelo de tarefa, análise de carga cognitiva, identificação de relacionamentos semânticos), um esquema de navegação entre *containers*, e seleciona componentes abstratos individuais para cada conceito de forma que sejam independentes de quaisquer ambientes de interação. Componentes abstratos individuais não são AIOs, pois são ainda mais abstratos que os AIOs na medida que não possuem atributos abstratos genéricos nem eventos abstratos.

Task & Concepts (T&C) descreve as várias tarefas a serem executadas e os conceitos orientados ao domínio requeridos para o desempenho destas tarefas.

~~*Task & Concepts (T&C) Tarefas e Conceitos: descreve as várias tarefas a serem executadas e os conceitos orientados ao domínio requeridos para o desempenho destas tarefas.*~~

O *Cameleon Reference Framework* prevê que os passos de desenvolvimento possam ser obtidos através de *transformações* verticais e horizontais. As transformações verticais definem os processos para a transformação de um passo de desenvolvimento - um modelo de IU - em outro mais concreto (processo de reificação), ou em outro mais abstrato (processo de abstração). As transformações horizontais definem processos para a obtenção de um modelo de IU a partir de outro de mesmo nível de reificação mas em um contexto de uso diferente do modelo original (processo de tradução).

As transformações verticais tanto podem ser processadas através de abordagens *top-down* (reificação) quanto *bottom-up* (abstração). A reificação compreende um processo

de derivação, onde se parte de modelos abstratos de mais alto nível até modelos de implementação (*run time*). De forma oposta, a abstração segue um processo de engenharia reversa tornando possível inferir modelos abstratos a partir de modelos mais concretos.

As transformações verticais não necessariamente têm como ponto de entrada (*entry point*) os passos de desenvolvimento mais extremos do CRF. Um ponto de entrada assinala um passo de desenvolvimento ao qual o processo de transformação (abstração e/ou tradução) pode ser iniciado. Como exemplo, uma AUI pode ser obtida sem que a T&C tenha nível de reificação ao qual o processo de transformação (abstração e/ou tradução) pode ser iniciado. Logo, o CRF prevê pontos de entrada em qualquer nível de reificação. Por exemplo: uma AUI pode ser obtida sem a T&C haver sido definida.

Para exemplificar um modelo de IU descrito pelo CRF, considere a representação da modelagem de um botão de *download* de arquivo apresentada na figura 4.6.

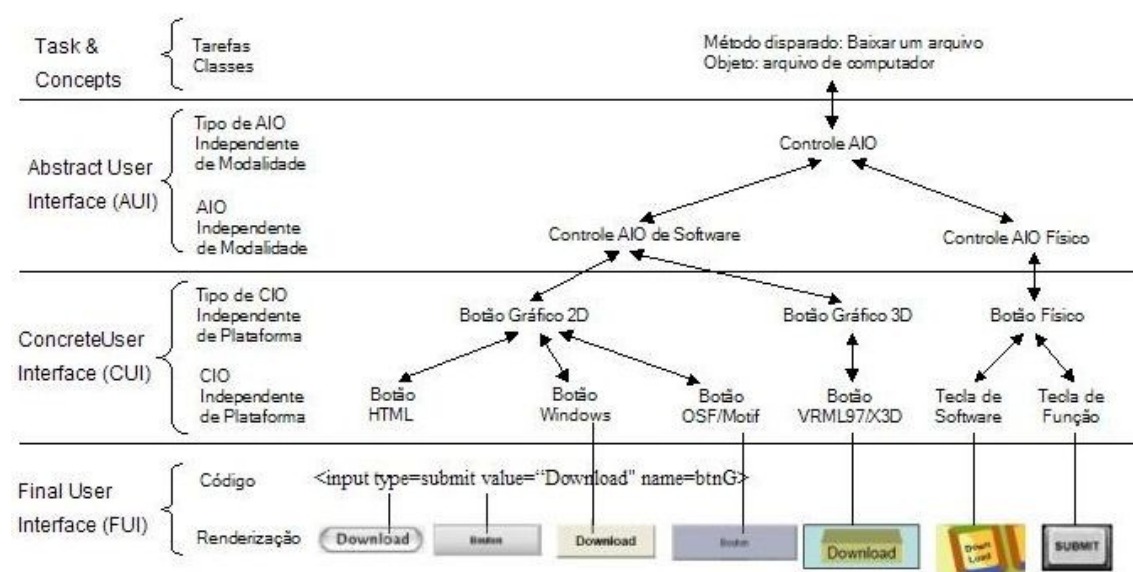


Figura 4.6: Exemplo de transformações no CRF.

Fonte: Vanderdonckt et al. 2004

No nível FUI, a Renderização materializa como uma IU, codificada em uma linguagem (de marcação, de programação ou declarativa), é renderizada dependendo do *toolkit* da IU, do gerenciador de janelas e do gerenciador de apresentação. Por exemplo, um botão de pressionamento (“*push button*”) programado em HTML no subnível de Código pode ser renderizado diferentemente em MacOS X e Java Swing. Portanto, o subnível de Código é materializado em subníveis de Renderização.

Considerando que o nível CUI é responsável por abstrair um nível FUI independente de qualquer plataforma computacional, este nível pode ser decomposto em dois subníveis: CIO Independente de Plataforma e Tipo de CIO Independente de Plataforma. Por exemplo, um botão de pressionamento em HTML pertence ao tipo “Botão de Pressionamento Gráfico 2D”. Outros membros desta mesma categoria incluem o Botão de Pressionamento do Windows e o `XmButton` do OSF/Motif.

Considerando que o nível AUI é responsável por abstrair o CUI independentemente de qualquer modalidade de interação, este nível pode ser decomposto em dois subníveis: AIO Independente de Modalidade e Tipo de AIO Independente de Modalidade. Por exemplo, um controle de software (seja ele 2D ou 3D) e um controle físico (ex: um

botão físico em um painel de controle ou uma tecla de função) pertencem à mesma categoria de Controle AIO.

No nível T&C, uma tarefa de um certo tipo (no exemplo: Baixar um arquivo) é especificada de tal forma que naturalmente conduza ao AIO com o controle da tarefa de baixar arquivo.

Graças aos quatro níveis de abstração, é possível estabelecer mapeamentos entre instâncias e objetos existentes nos diferentes níveis de abstração e desenvolver transformações que realizem abstrações, reificações ou traduções. Por exemplo, se uma interface gráfica (GUI) precisa ser virtualizada, uma série de abstrações podem ser aplicadas até que o subnível “Controle de Software AIO” seja alcançado. Então, uma série de reificações é aplicada para voltar para o nível FUI de forma a obter um objeto que atenda às mesmas necessidades que o anterior, só que, agora, em 3D.

4.2.4 Linguagens de descrição de interfaces com o usuário

Com objetivo análogo ao dos modelos de interface de usuário, foi criado o conceito de Linguagens de Descrição de Interfaces com o Usuário (*User Interface Description Languages – UIDLs*). Essas linguagens têm como objetivo a descrição da interface de usuário em alto nível através de um código interpretável, o qual pode ser mapeado para diferentes plataformas.

Uma linguagem para a descrição de interfaces deve atender alguns requisitos mínimos, especialmente quando se tem como objetivo o reuso de IUs para múltiplas plataformas.

Inicialmente, a linguagem deve fornecer uma separação entre o código da lógica da aplicação e o código da interface. Sem esta separação, torna-se mais difícil, por exemplo, o uso de uma mesma interface por diferentes dispositivos, uma vez que é comum que cada plataforma tenha a sua própria linguagem de programação para a lógica da aplicação.

A linguagem também precisa ser portátil, isto é, ela deve ser capaz de ser entendida por diferentes dispositivos. Esta característica é necessária para que uma interface descrita em uma linguagem possa ser entendida e manipulada em diversas plataformas.

Além disto, é importante que a linguagem descreva de maneira concisa, organizada e separada os diversos elementos que compõem uma interface. Em especial, os seguintes elementos são importantes em uma descrição de interface [Phanouriou 2000]
~~uma interface. Em especial, os seguintes elementos são importantes em uma descrição de interface:~~

- *Estrutura*: a linguagem deve fornecer uma descrição abstrata dos objetos que a compõem, como botões e caixas de texto. Esta descrição deve ser abstrata o suficiente para que os objetos possam ser representados corretamente em múltiplas plataformas e diferentes metáforas de interface, como interfaces gráficas e de voz.
- *Estilo*: a linguagem deve permitir a descrição do estilo de seus objetos, como tamanho, cor e disposição. Esta descrição deve ser feita de forma a permitir a fácil adaptação da interface a diferentes contextos de uso. Ou seja, a alteração de estilo dos objetos deve ser facilitada em tempo de execução. Também é

importante que a linguagem descreva como estes objetos estão organizados e de que forma eles se relacionam para formar a interface.

- *Conteúdo*: a linguagem deve ser capaz de separar o conteúdo (textos, sons, imagens) dos outros elementos da interface. Com isto, é possível alterar o conteúdo da interface sem mudar a sua estrutura ou estilo. Esta característica facilita aspectos como a internacionalização (isto é, o suporte a diferentes idiomas).
- *Comportamento*: a linguagem deve ser capaz de descrever como a interface reage às interações do usuário. Isto pode ser feito, por exemplo, relacionando uma ação a cada objeto descrito na estrutura da interface.
- *Lógica*: a linguagem deve permitir a conexão da interface com a lógica da aplicação. Ou seja, ela deve descrever de que forma as reações do usuário disparam os procedimentos internos da aplicação que utiliza a interface.
- *Apresentação*: a linguagem deve fazer a correspondência entre a descrição da interface e o dispositivo. Uma maneira de realizar esta tarefa é fazendo um mapeamento direto entre os objetos abstratos descritos na estrutura da interface com os objetos concretos do dispositivo alvo.

Ainda que nem todos estes elementos sejam estritamente necessários em uma linguagem para descrição de interfaces, é desejável que todos eles estejam, pelo menos, representados de alguma forma na linguagem.

Um outro requisito desejável é que a linguagem seja de fácil compreensão não apenas para programadores, mas também para projetistas de interfaces que não tenham muita experiência com programação. Desta forma, quanto mais simples for a linguagem, mais facilmente ela será compreendida pelo projetista de interfaces que a estiver utilizando.

Com base nestes requisitos, houve uma convergência de propostas de UIDLs baseadas na recomendação *eXtensible Markup Language* (XML) [\[Cover Pages 2005, Souchon & Vanderdonckt 2003\]](#). Por ser uma linguagem de marca, a XML pode ser entendida por qualquer outra linguagem de programação ou ferramenta de desenvolvimento. Ou seja, ela pode ser utilizada em qualquer plataforma. Devido à forma hierárquica como a XML é constituída, ela fornece uma estrutura ideal para a separação e organização dos diversos elementos que compõem uma interface.

Desta forma, uma UIDL é uma linguagem baseada em XML que permite descrever as IUs de forma independente de tecnologia e de ambiente de desenvolvimento integrado (IDE).

Atualmente, existem várias propostas de UIDLs, na sua maioria com o propósito de descrever IUs para ambientes de interação e/ou plataformas específicos, dentre as quais podemos citar: AAIML [\[Zimmermann et al. 2002\]](#) para *Consoles Remotos Universais (URCs)*, AUIML [\[Arhelger et al. 2004\]](#) para *Java Swing e Web*, MXML [\[Coenraets 2004\]](#) para o *Macromedia Flash*, SEESCOA XML [\[SEESCOA 2002\]](#) para *Java*, XAML [\[XAML 2007\]](#) para o *sistema operacional Windows Longhorn*, e XForms [\[Forms 2007\]](#), XICL [\[XICL 2005\]](#) e XUL [\[XUL 2007\]](#) para *interfaces Web*. Entretanto, existem pelo menos 3 propostas com o objetivo de descrever IUs em uma variedade maior de ambientes de interação e que, portanto, são mais adequadas para abordagens de reuso de IUs como a que está sendo proposta neste trabalho. Basicamente, estas abordagens são:

~~XIML, UIML e UsiX para Consoles Remotos Universais — (URCs), AUIML para Java Swing e Web, MXML para o Macromedia Flash, SEESCOA XML para Java, XAML para o sistema operacional Windows Longhorn, e XForms, XICL e XUL para interfaces Web. Entretanto, existem pelo menos 3 propostas com o objetivo de descrever IUs em uma variedade maior de ambientes de interação e que, portanto, são mais adequadas para abordagens de reuso de IUs como a que está sendo proposta neste trabalho. Basicamente, estas abordagens são: UIML, UsiXML e XIML, as quais são descritas a seguir.~~

4.2.4.1 XIML

A linguagem XIML (*Extensible Interface Markup Language*) [Puerta & Eisenstein 2002] foi padronizada pelo XIML Forum [XIML Forum 2007]. ~~O XIML Forum é uma organização que se propõe à pesquisa, disseminação, adoção e padronização da XIML como uma linguagem para a descrição de interfaces. Através da XIML, o Forum pretende fornecer uma especificação comum e uma infra-estrutura de desenvolvimento para todos os tipos de desenvolvedores de interfaces e se propõe a representar interfaces permitindo o suporte universal à funcionalidade através de todo o ciclo de vida de uma interface. As fases previstas neste ciclo são: design, desenvolvimento, operação, administração, organização e avaliação.~~

A XIML se propõe a representar interfaces permitindo o suporte universal à funcionalidade através de todo o ciclo de vida de uma interface. As fases previstas neste ciclo são: design, desenvolvimento, operação, administração, organização e avaliação.

Os principais requisitos de design da linguagem XIML são:

- Ter um repositório central de dados.
- Suportar todos os ciclos de vida de uma interface.
- Ter elementos concretos e abstratos.
- Oferecer suporte às relações entre os elementos de uma interface.
- Ser baseado em uma tecnologia simples (a linguagem XML).

A linguagem XIML descreve uma interface através de uma coleção organizada de elementos. Estes elementos, por sua vez, são categorizados por um ou mais componentes. Em sua versão inicial, a XIML define cinco componentes básicos:

- *Tarefa*: define hierarquicamente os processos de interação e as tarefas de usuário suportadas pela interface.
- *Domínio*: define hierarquicamente objetos de interface e classes de objetos.
- *Usuário*: define uma hierarquia de usuários da interface.
- *Apresentação*: define uma hierarquia de elementos de interação que representam os objetos concretos da interface com os quais o usuário interage.
- *Diálogo*: define uma coleção estruturada de elementos que determinam as ações de interação que estão disponíveis aos usuários da interface.

~~O grupo que define a linguagem, o XIML Forum, é uma organização que se propõe à pesquisa, disseminação, adoção e padronização da XIML como uma linguagem para a descrição de interfaces. Através da XIML, o Forum pretende fornecer uma~~

~~especificação comum e uma infra-estrutura de desenvolvimento para todos os tipos de desenvolvedores de interfaces. O XML Forum também procura financiar pesquisas na área de Interação Homem-Computador (IHC), especialmente no estudo de tecnologias avançadas de interfaces com o usuário.~~

A principal dificuldade em se estudar a XIML é que praticamente todas as informações (incluindo a especificação da linguagem) são disponibilizadas apenas para membros do XML Forum. Para participar deste fórum, deve ser aceita uma licença que proíbe a ampla disseminação de qualquer informação contida no fórum.

4.2.4.2 UIML

A UIML é uma meta-linguagem para a descrição de IUs. A UIML (User Interface Markup Language) [Abrams et al. 1999, UIML 2004] é uma linguagem de marcação semelhante ao HTML. ~~A UIML (User Interface Markup Language) [Abrams et al. 1999, UIML 2004] é uma linguagem de marcação semelhante ao HTML~~ com o principal objetivo de servir como uma forma canônica (única) para a descrição de interfaces e com a proposta de se tornar um padrão de descrição de interfaces [Luyten & Coninx 2004].

Desta forma, qualquer ferramenta de desenvolvimento de interfaces pode armazenar a sua interface em UIML, para que depois qualquer outra ferramenta de desenvolvimento possa utilizar esta interface, possibilitando o desenvolvimento de interfaces independentes de ferramentas de desenvolvimento. Para isto, basta que as duas ferramentas sejam capazes de entender a linguagem.

Para isto, um grupo de desenvolvedores de interfaces do Instituto Politécnico e Universidade Estadual da Virginia (Virginia Tech) [CHCI 2007] definiu um conjunto de requisitos para a criação da UIML:

- Permitir o desenvolvimento de interfaces para qualquer plataforma sem a necessidade de se aprender as linguagens de programação específicas de cada ambiente.
- Reduzir o tempo necessário para o desenvolvimento de interfaces para uma família de plataformas.
- Fornecer uma separação natural entre o código da interface e o código da lógica da aplicação.
- Permitir a implementação de interfaces por pessoas sem muita experiência com programação.
- Permitir a rápida prototipação de interfaces.
- Simplificar o desenvolvimento de interfaces para múltiplos idiomas.
- Permitir o *download* eficiente de interfaces para plataformas clientes através de redes de computadores como a Internet.
- Permitir a extensão para o suporte a futuras tecnologias de interface.

A UIML descreve uma interface separando-a em seis partes que podem ser definidas respondendo-se às seguintes perguntas:

- Quais são os elementos que compõem a estrutura da interface?

- Qual é o estilo de cada elemento da interface (tamanho, cor)?
- Qual é o conteúdo associado a cada elemento da interface (textos, sons, imagens)?
- Qual é o comportamento da interface frente às interações do usuário?
- Como é feita a conexão com a lógica da aplicação?
- Como a apresentação é mapeada para uma metáfora de interface?

Para melhor definir esta separação, foi criada uma extensão do modelo *Model-View-Controller* (MVC), chamado *Meta-Interface Model* (MIM) [Phanouriou 2000]. No modelo MIM, a interface é separada em três componentes principais que a descrevem de forma canônica e independente de plataforma: Interface, Apresentação e Lógica. Estes componentes e como eles se relacionam são apresentados na figura 4.7~~se relacionam são apresentados na figura 4.6.~~

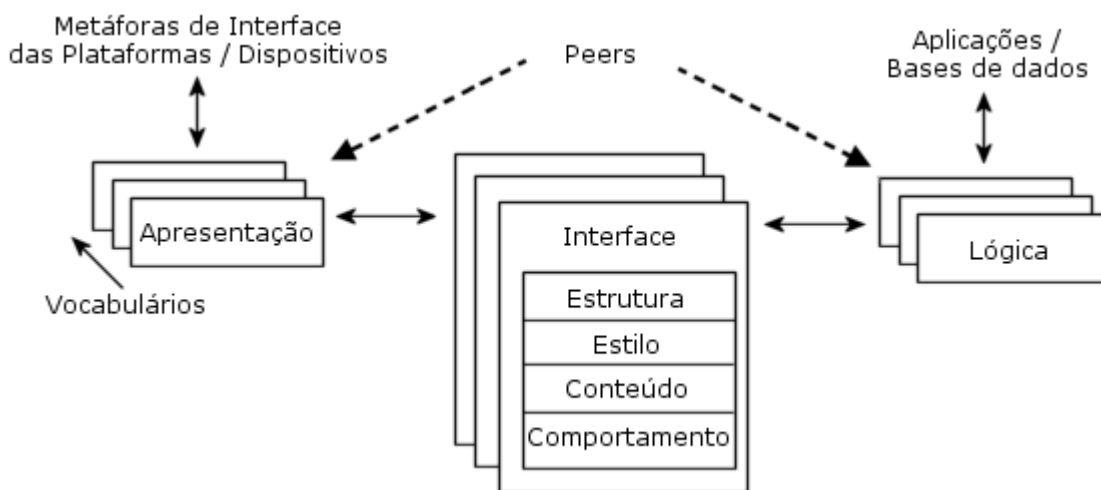


Figura 4.76: O Modelo MIM.

A componente de Interface contém a descrição da IU e descreve como é feita a interação entre o usuário e a interface, utilizando, para isto, um conjunto de objetos, eventos e chamadas de métodos abstratos. Esta componente é ainda subdividida em quatro componentes, cada um descrevendo um aspecto de como cada objeto contribui para a interface: a *Estrutura* descreve a organização dos objetos de interação na interface; o *Estilo* descreve as propriedades específicas para a apresentação de cada objeto, fazendo um mapeamento entre os objetos abstratos e os concretos; o *Conteúdo* descreve as informações externas que a interface deve apresentar ao usuário, como textos e sons; o *Comportamento* descreve as interações entre o usuário e os objetos em tempo de execução, enumerando um conjunto de condições e as ações associadas.

O componente de *Lógica* determina como a interface é conectada à lógica da aplicação, descrevendo o mapeamento de eventos na interface com funções do aplicativo sendo executado.

No componente de *Apresentação* está especificado o vocabulário que será utilizado para mapear a definição da interface, ou seja, como a interface descrita na linguagem UIML será apresentada ao usuário. Dessa forma, a tecnologia utilizada pela interface pode ser facilmente modificada, devendo-se apenas substituir o vocabulário utilizado.

4.2.4.3 UsiXML

A linguagem UsiXML (*User Interface Extensible Markup Language*) [UsiXML 2006] tem como objetivo descrever interfaces para múltiplos contextos de uso através de diferentes níveis de detalhamento e abstração. Ou seja, aplicações com diferentes tipos de técnicas de interação, modos de uso e plataformas podem ser descritas de maneira a preservar o seu design. A linguagem propõe uma integração de diversas outras linguagens já existentes, tais como cHTML, WML, HTML, XHTML, VoiceXML, VRML, Java e C++.

A UsiXML consiste de uma linguagem declarativa que captura a essência da uma interface independente das características físicas. Isto é feito através de uma descrição abstrata dos elementos que constituem uma interface. Estes elementos são agrupados em diversos modelos, como o de interface, o de mapeamento, o de domínio, o de tarefa, o de contexto, e o de mapeamento, de domínio, de tarefa, de contexto, de transformação, entre outros.

A linguagem suporta o desenvolvimento de interfaces independentes de plataforma. Caso seja necessário, a UsiXML permite que sejam incorporadas à descrição da interface referências a uma plataforma específica. Também é possível o reuso de elementos descritos em um UsiXML para a criação de uma nova interface UsiXML.

4.3 Abordagens para reuso de IHC

Embora o reuso seja uma das práticas mais indicadas da Engenharia de Software, a sua aplicação em IHC é ainda pouco difundida, talvez pela crença muito presente entre os designers de que reuso de soluções de outrem é um sinal da diminuição da sua criatividade. Sua adoção na prática de design de interação por profissionais de IHC potencialmente permitiria uma maior qualidade deste design (reuso de uma solução bem sucedida em teoria diminui a probabilidade de ocorrência de erros) e uma maior produtividade da equipe (reuso de elementos libera os designers para tratar problemas para os quais ainda não há solução)~~(reuso de elementos libera os designers para tratar problemas para os quais ainda não há solução) da equipe.~~

~~Mas a adoção de práticas de reuso não é implantada facilmente em uma organização. Construir artefatos reusáveis requer informações de identificação, extração, organização e representação de uma maneira que seja fácil de entender e manipular. Encontrar os artefatos reusáveis que possam ser (re)utilizados para o desenvolvimento de um novo sistema pode ser muitas vezes mais difícil e trabalhoso do que o desenvolvimento deste novo sistema. De fato, software para ser reusável e reusado deve ser projetado, documentado e implementado para este fim de reuso [Traez 1994].~~

~~Embora a tecnologia disponível já habilite os designers a praticar reuso, isto não implica que o reuso ocorrerá. É preciso uma estratégia de reuso (com conjunto de conceitos e ferramentas associados) que propicie o reuso em diferentes níveis de abstração. Tal estratégia deve focar em reuso durante todo o ciclo de desenvolvimento, prover suporte à reutilização de artefatos (*development with reuse*) e à produção de artefatos reusáveis (*development for reuse*), ser facilmente integrável a outros métodos e técnicas de desenvolvimento utilizados e ser implementada por meio de ferramentas (também integráveis a ferramentas correntemente utilizadas) [Sindre et al. 1995].~~

A seguir, são apresentados os principais métodos e técnicas que, atualmente, podem contribuir para o reuso de interfaces com o usuário. Cada proposta será apresentada

separadamente e será feita uma análise da sua aplicabilidade nas atividades de projeto de IU em um processo de desenvolvimento OO.

4.3.1 **Recomendações e guias de estilo**

Para atingir o objetivo de obter um sistema com usabilidade, a abordagem típica e mais utilizada em IHC é seguir *recomendações (guidelines)* [Brown 1988, Cooper & Reimann 2003, Mayhew 1992, Nielsen 1999, Nielsen & Loranger 2006] e *guias de estilo (style guides)* [GNOME 2004, IBM 1992, KDE 2007, Microsoft 1999, Sun 2002, Trower 1995].

Recomendações têm o propósito de captar o conhecimento do projeto de interfaces em pequenas regras, as quais podem ser usadas para construir novas interfaces com o usuário.

Os guias de estilo são recomendações de IU que têm o objetivo de definir padrões de IU (*look-and-feel*) para sistemas que são concebidos para rodar em determinadas plataformas. Desta forma, o projetista de IU que segue e implementa as recomendações do guia de estilo estará construindo IUs consistentes (tanto na aparência quanto no comportamento) com as dos demais sistemas da mesma plataforma. Com isso, os usuários aprenderão a utilizar o sistema de forma mais rápida, pois os elementos da interface aparentam e se comportam da forma com que estão acostumados a usá-los. Já faz tempo que as recomendações vêm sendo usadas para obter conhecimento de projeto de interfaces e ajudar projetistas a (re)usar esse conhecimento quando projetam as interfaces. O conhecimento destas recomendações ajuda o projetista de interfaces a tomar as decisões corretas e minimiza a possibilidade de cometer os mesmos erros.

Já faz tempo que as recomendações são usadas para obter conhecimento de projeto de interfaces e ajudar projetistas a (re)usar esse conhecimento quando projetam as interfaces. O conhecimento destas recomendações ajuda o projetista de interfaces a tomar as decisões corretas e minimiza a possibilidade de cometer os mesmos erros.

Entretanto, a aplicação das recomendações não está livre de problemas. van Welie enfatiza que muitas vezes as recomendações são muito simplistas ou muito abstratas. Como são muito numerosas, torna-se difícil escolher qual delas se aplica a um problema em particular. Além disso, as recomendações podem conflitar umas com as outras e, conseqüentemente, o projetista pode não conseguir resolver o problema de projeto da interface [Welie 2001].

Uma das razões para estes problemas é que a maioria das recomendações sugere uma validade absoluta quando, na verdade, a sua aplicabilidade depende de um contexto. Este contexto é crucial para saber que recomendações usar, porquê e como elas devem ser “traduzidas” para um contexto específico. Para muitas decisões de projeto, é necessário conhecer o contexto de uso (incluindo aí as tarefas, os usuários, o ambiente, etc). Sem este conhecimento, o problema de projeto de interfaces não pode ser resolvido adequadamente. As recomendações não têm uma forma explícita de declarar o contexto ou, quando muito, ele é brevemente mencionado [Welie 2001].

4.3.2 Padrões de interação e linguagens de padrões

Os conceitos de *padrões (patterns)* e de *linguagens de padrões* são derivados da Arquitetura e foram propostos por Christopher Alexander et al. em [Alexander et al. 1977]. Entretanto, os padrões começaram a se tornar populares na área de TI com a larga aceitação do livro “Padrões de Projeto – Soluções reutilizáveis de Software Orientado a Objetos” de Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (conhecidos como GoF - Gang of Four) [Gamma et al. 2000].

Os primeiros textos relacionados especificamente ao desenvolvimento de UI associados a padrões de interação (*interaction patterns* - IP) começaram a surgir a partir de 1994 [Rijken 1994]. Entretanto, somente a partir dos trabalhos de [Borchers 1999, 2000], [Sutcliffe & Carroll 1999] e [Tidwell 1998] é que se deu um incremento da aplicação de padrões na área de Design de Interação. Estas coleções de padrões foram criadas com o objetivo de melhorar o reuso de IHC.

Um padrão de interação deve captar conhecimentos *comprovados* de projeto de interfaces e é descrito em termos de um problema, um contexto e uma solução. Para um padrão, é importante que a solução seja uma solução comprovada para o problema declarado e os projetistas concordem com o fato de que esta é uma solução comprovada. Um padrão de interação deve estar focado em soluções que melhorem a usabilidade do sistema em uso [Welie et al. 2000].

Em comparação com as recomendações, os padrões contêm conhecimentos de projeto muito mais complexos, de forma que é comum que várias recomendações estejam integradas em um único padrão [Welie et al. 2000].

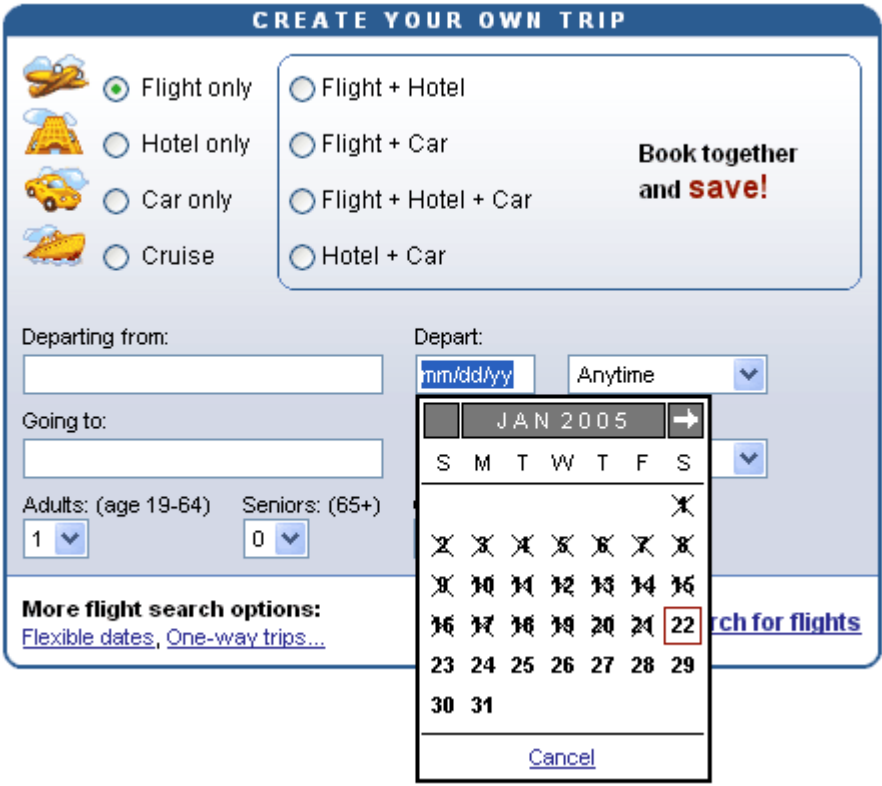
Os principais elementos de descrição de um padrão de projeto também podem ser usados na descrição de padrões de interação. Entretanto, van Welie observa que esses elementos devem ser descritos do ponto-de-vista “correto” [Welie et al. 2000], a saber:

- *Problema*: Problemas em padrões de interação devem ser problemas de usabilidade do sistema em uso. Os problemas devem ter relação com o uso do sistema e devem ser relevantes para o usuário ou qualquer outro stakeholder com interesse em usabilidade. Em contraste aos padrões de projeto, as descrições dos problemas nos padrões de interação não devem se focar nos problemas de construção que os projetistas se deparam, mas ser orientadas às tarefas dos usuários.
- *Contexto*: O contexto também é focado no usuário. Quais são as características do contexto de uso, considerando as tarefas, usuários e ambiente para o qual o padrão pode ser aplicado?
- *Solução*: Uma solução deve ser descrita muito concretamente e não deve impor novos problemas. Entretanto, uma solução descreve somente a solução núcleo e outros padrões podem ser necessários para resolver subproblemas. Outros padrões relevantes para a solução também devem ser referenciados.
- *Exemplos*: O exemplo deve apresentar como o padrão foi usado com êxito em um sistema. Muitas vezes, um exemplo pode ser dado usando uma screenshot e um texto adicional que explica o contexto daquela solução em particular. É preferível usar exemplos de sistemas reais de forma que a validade do padrão seja reforçada. Se o redator do padrão não consegue encontrar nenhum

exemplo na vida real, ou o padrão não é um bom padrão ou é raramente utilizado (ou ambos).

Um exemplo bem simples de um padrão de interação é apresentado na tabela 4.2. O padrão é referente a um problema que um usuário possa ter, como ele pode ser resolvido e porque ele funciona.

Tabela 4.2: Um exemplo de Padrão de Interação
Fonte: van Welie 2005

Nome	<p>Seletor de data</p>  <p>De www.expedia.com</p>
Problema	Usuários precisam selecionar uma data ou um horário
Use quando	<p>O usuário deve selecionar uma data em um formulário. Por exemplo, para selecionar a data de início ou de fim de uma Inscrição ou para preencher uma data de nascimento em um Formulário de cadastramento. Uma data também pode ser usada em um formulário de Pesquisa Avançada onde os usuários podem buscar por itens antes/depois de uma data específica. Um uso típico é na parte de uma aplicação para encontrar horários de partida/chegada de aviões, navios ou trens.</p> <p>Quando os usuários precisam selecionar um período de tempo horário, dois seletores de data podem ser usados: um para a data de início e outro para a data de término.</p>
Solução	<p>Use uma combinação de uma caixa de edição e um calendário gráfico clicável.</p> <p>Para usuários experientes, uma caixa de edição onde a data pode ser digitada é, de longe, a opção mais rápida para informar uma data. Todavia, uma dica deve ser colocada ao lado ou abaixo da caixa de edição para indicar o formato esperado. Depois que a caixa de edição perde o seu foco, a data digitada deve ser verificada e reformatada. É recomendado permitir aos usuários usar múltiplos tipos de separadores, tais como '/' e '-' bem como anos de 2 ou 4 dígitos.</p> <p>Para usuários que preferem uma interface mais gráfica ou não sabem a data exata, um calendário mensal clicável é apresentado. Dias são apresentados em uma tabela com linhas de semanas. No topo da tabela, um seletor de mês é apresentado de forma que o usuário possa ir para o próximo mês ou para o mês anterior, ou pular para um mês em</p>

específico usando uma caixa de combinação. Quando o usuário clica em um dos dias do calendário, então aquela data será a digitada no campo de edição.

Em muitos casos, os usuários só podem selecionar um número particular de dias tal como, por exemplo, no caso de reservas de hotel onde somente em determinadas datas existem quartos disponíveis. Isto pode ser mostrado no seletor de datas fazendo com que somente as datas “possíveis” possam ser clicáveis e diferenciadas visualmente.

Opcionalmente, também pode haver um link para voltar o calendário para o dia presente. Quando é necessário selecionar semanas, então pode existir um ícone próximo das linhas das semanas para selecionar semanas inteiras.

Por quê

Usuários já estão muito familiarizados com calendários mensais. Este calendário lhes permite selecionar uma data rapidamente e previne que informem uma data inexistente. Ao combinar um calendário gráfico com uma caixa de edição, usuários que já estão acostumados a informar datas, podem informá-las eficientemente. Estas soluções combinam a clareza de um calendário gráfico com a velocidade da digitação via teclado.

Mais exemplos

Step 1: Search flight 1 2 3 4 5
Online booking in 5 steps

☒ Return ☐ One way

From: Amsterdam ▼ To: Innsbruck ▼

Going out: 19 ▼ March 2005 ▼ Coming back: 26 ▼ March 2005 ▼

wk	Mo	Tu	We	Th	Fr	Sa	Su
9		1	2	3	4	5	6
10	7	8	9	10	11	12	13
11	14	15	16	17	18	19	20
12	21	22	23	24	25	26	27
13	28	29	30	31			

wk	Mo	Tu	We	Th	Fr	Sa	Su
9		1	2	3	4	5	6
10	7	8	9	10	11	12	13
11	14	15	16	17	18	19	20
12	21	22	23	24	25	26	27
13	28	29	30	31			

Number of passengers

1 ▼ Adults and children from 12 yrs

0 ▼ Children 2 to 11 yrs

0 ▼ Infants 0 to 23 months

Search flights ►

Em www.transavia.com o seletor de datas não é um pop-up, mas é diretamente apresentado abaixo da caixa de edição da data. Uma vez que os aeroportos “De” e “Para” foram selecionados os usuários podem ver em quais datas existem vôos por companhia aérea, o que os permite ajustar suas datas.

Em www.klm.com é encontrada uma forma ligeiramente diferente, mas também comum de usar um seletor de datas. Os dias e meses estão em caixas de combinação, mas também existe um ícone próximo delas que permite abrir o seletor de data.

Em www.baylor.com uma variação é usada onde os usuários também podem selecionar uma semana inteira num [Calendário de Eventos](#).

Sutcliffe e Carroll, em [Sutcliffe & Carroll 1999], fazem uma proposta um pouco diferente para descrever padrões. Eles usam a abordagem de *afirmações (claims)*. Apesar de usarem outra terminologia, a estrutura é bem semelhante. Entretanto, não existe uma descrição explícita do problema nem uma especificação de contexto.

Os padrões raramente existem de forma isolada. Os autores de padrões, geralmente os colocam em coleções, geralmente conhecida como *linguagens de padrões (pattern languages)*.

As linguagens de padrões possuem padrões que se relacionam e se complementam entre si, disponibilizando para o projetista de interfaces um acervo de idéias comprovadas de interação que podem ser aplicadas no seu projeto de forma consistente.

Entretanto, as linguagens de padrões ainda não são utilizadas de forma sistemática nos projetos de interface. [Gaffar et al. 2004] salientam que o mero fato de existir uma multiplicidade destas linguagens de padrões e a incompatibilidade entre elas parece ser parte deste problema.

Mahemoff reforça que uma *linguagem de padrões* é constituída quando uma coleção de padrões é arranjada em uma rede de padrões interdependentes, especialmente onde padrões de mais alto nível rendem contextos os quais são resolvidos por padrões mais detalhados [Mahemoff & Johnston 2001]. Em 2001, Mahemoff já havia identificado pelo menos 12 propostas de coleções de padrões ou de linguagens de padrões na área de UI (além da própria linguagem de padrões proposta por ele). Um resumo destas propostas é apresentado na tabela 4.3 onde foram acrescentados mais algumas propostas que surgiram após 2001.

Em 2001, Mahemoff já havia identificado pelo menos 12 propostas de coleções de padrões ou de linguagens de padrões na área de UI (além da própria linguagem de padrões proposta por ele) [Mahemoff & Johnston 2001]. Um resumo destas propostas é apresentado na tabela 4.3 onde foram acrescentados mais algumas propostas que surgiram após 2001.

Tabela 4.3: Um survey de coleções de padrões de IHC

~~Fonte: Mahemoff & Johnston 2001~~

Abordagem	Nível de abstração	Mídia Alvo	Requisitos Especializados
[Tidwell 1998]: Interaction Design <i>Patterns</i>	Sistemas, Elementos de UI Múltiplos e Singulares, Funcionalidade	Aplicações GUI, Websites	Nenhum
[Brighton 2001]: Brighton Usability <i>Pattern</i> Collection	Sistemas completos, Elementos de UI Múltiplos e Singulares, Funcionalidade	Aplicações GUI	Nenhum
[Welie & Trætteberg 2000]: Amsterdam <i>Pattern</i> Collection	Elementos de UI Múltiplos, Funcionalidade	Aplicações GUI, Websites	Nenhum
[Coram & Lee 1996]: Experiences	Elementos de UI Múltiplos e Singulares, Funcionalidade	Aplicações GUI	Nenhum
[Wake 1998]: <i>Patterns</i> for Interactive Application	Elementos de UI Múltiplos e Singulares, Funcionalidade	Aplicações GUI	Nenhum
[Borchers 1999]: Interdisciplinary Design <i>Patterns</i>	Tanto de Alto-nível quanto de baixo-nível, Funcionalidade	Vários	Pode ser específico a um domínio

Abordagem	Nível de abstração	Mídia Alvo	Requisitos Especializados
[Cybulski & Linden 2000]: <i>Multimedia Patterns</i>	Elementos de UI Múltiplos e Singulares, Funcionalidade	Aplicações multimídia	Nenhum
[Bradac 1998]: <i>Patterns for Form Style Windows</i>	Elementos de UI Múltiplos	GUI Forms	Nenhum
[Perzel & Kane 1999]: <i>Usability Patterns for Applications on the World Wide Web</i>	Elementos de UI Múltiplos e Singulares, Funcionalidade	Websites	Nenhum
[Riehle & Zullighoven 1995]: <i>Tool Construction and Integration</i>	Elementos de UI Múltiplos, Funcionalidade, Projeto de Software	Aplicações Desktop	Manipulação de Artefatos
[Breedvelt-Schouten et al. 1997]: <i>Reusable Structures in Task Models</i>	Tarefas	Vários	Nenhum
[Stimmel 1999]: <i>Patterns for Developing Prototypes</i>	Processo de desenvolvimento	Vários	Nenhum
[Mahemoff 2001]: <i>Planet Patterns</i>	Processo de desenvolvimento, Especificação de Alto-nível, Projeto de Software	Vários	Internacionalização de Software
[Tidwell 2005]: <i>UI Patterns and Techniques</i>	Elementos de UI Múltiplos e Singulares, Funcionalidade	Aplicações GUI, Websites	Nenhum
[Welie 2005]: <i>patterns in Interaction Design</i>	Elementos de UI Múltiplos, Funcionalidade	Aplicações GUI, Websites	Nenhum
[Yahoo 2006]: <i>Design Pattern Library</i>	Elementos de UI Múltiplos	Websites	Nenhum

Todas as linguagens de padrões e coleções de padrões de interação existentes têm em comum que o seu uso está focado nas fases de projeto de interfaces e avaliação de uso. Infelizmente, elas ainda não estão integradas a outras atividades do ciclo de vida de desenvolvimento do software tais como a análise de requisitos [Metzker & Reiterer 2002].

4.3.3 Geradores de interface

Automatizar a geração de código de computador tem sido uma das mais antigas iniciativas para aumentar a produtividade do desenvolvimento de sistemas.

Para se gerar código, o gerador precisa se basear em modelos formais que forneçam a “matéria-prima” do código a ser gerado [Kleppe et al. 2004]. Estes modelos servem de entrada para o gerador que, com base na sua especificação e na definição da linguagem-alvo “gera” um código novo.

Devido à complexidade em si do projeto de interfaces do usuário e da inexistência de um modelo formal para a especificação destas interfaces, a utilização de geradores de código para produzir as interfaces com o usuário ainda não é uma realidade.

Como a maioria dos programas geradores de código existentes atualmente, a entrada para a geração das interfaces é uma especificação de tabela ou de um conjunto de tabelas de um banco de dados relacional. De posse destas especificações, o gerador cria interfaces de programas do tipo CRUD com interfaces geralmente espartanas do ponto de vista da usabilidade.

4.3.4 Componentização

Os componentes de interface são a forma mais utilizada de reuso de interfaces. Hoje em dia, praticamente a totalidade das ferramentas IDE disponíveis oferecem uma “paleta” de objetos de interface de usuário (*widgets*) que nada mais são do que os próprios componentes de interface. São exemplos de componentes de interface: botões, menus, caixas de texto, barras de rolagem, caixas de listagem, etc.

O programador seleciona o *widget* que lhe interessa e o arrasta para uma superfície que represente o contexto da interação que está sendo modelada. A partir daí, altera as características deste objeto de forma a deixá-lo com o aspecto e comportamento esperados. Muitas destas características podem ser alteradas de forma gráfica (tais como a altura, largura e posição de um botão) fazendo com que o processo de programação seja mais visual do que descritivo. Esta forma de programação visual, atualmente, é a forma padrão de desenvolvimento das interfaces com o usuário.

Um dos maiores problemas no uso de componentes de interface é que, geralmente, as ferramentas IDE disponibilizam apenas os componentes de interação mais elementares. Isto limita o reuso aos elementos mais básicos da interação.

Para incorporar componentes de interação mais complexos (compostos) à paleta de *widgets*, primeiro é necessário definir o padrão de interação adequado à solução do problema. Depois disto, pode-se pesquisar se já existe a disponibilidade de uma implementação deste padrão ou se é necessário desenvolvê-lo.

A pesquisa dos componentes disponíveis não é uma tarefa trivial, pois é necessário fazer uma pesquisa em várias fontes e consultar vários fornecedores. Questões de compatibilidade e qualidade dos componentes passam a ser críticas [Ravichandran & Rothenberger 2003]. Além disso, o número de opções disponíveis para um determinado componente pode ser muito alto, fazendo com que o processo de prospecção e de seleção de componentes se torne uma tarefa não-trivial.

5 ABORDAGEM PROPOSTA

Vimos, nos capítulos anteriores, uma revisão das diversas práticas de reuso - em especial, das práticas que corroboram para o reuso de IHC - e dos principais aspectos envolvidos no projeto de IUs em um processo de desenvolvimento de software baseado na UML e dirigido por casos de uso.

A efetiva execução destas práticas tem contribuído significativamente para a melhoria da qualidade do software produzido e na redução dos esforços de desenvolvimento. Em particular, a adoção de padrões de interação como fonte para o projeto de IUs, tem contribuído para o aumento da usabilidade do software e, conseqüentemente, da percepção de valor do software e da satisfação em usá-los pelos usuários finais.

Também vimos que o cenário atual da prática de reuso ainda não atingiu um nível satisfatório que a incorporasse como prática rotineira e indispensável no processo de desenvolvimento do software. Segundo [Yongbeom & Stohr 1998], os principais aspectos que impedem o maior reuso são - além dos aspectos técnicos - os comportamentais, psicológicos, gerenciais e econômicos (ver seção 2.3). Entretanto, eliminar (ou mitigar) os aspectos de impedimento técnico (os que definem o processo e tecnologias de reuso) é fundamental para que uma boa prática de reuso se instaure, pois se não se sabe o quê, quando e como reusar, dificilmente investimentos nos demais aspectos compensarão a falta de um (bom) processo e tecnologia para reuso.

No aspecto técnico, ainda existem amplos espaços a serem preenchidos de forma a se chegar a uma metodologia eficaz para o desenvolvimento de software *com reuso e para reuso*.

A abordagem que será apresentada no decorrer deste capítulo, pretende dar uma contribuição no sentido de identificar, produzir e consumir artefatos reusáveis, com ênfase no reuso de IHC, em um processo de desenvolvimento dirigido por casos de uso.

O objetivo deste capítulo é apresentar a nossa abordagem para reuso de interfaces com o usuário (IUs) através da especificação e utilização de modelos de reificação de padrões de interação e de padrões de casos de uso, os quais denominamos: *padrões concretos de interação (Concrete Interaction Patterns – CIPs)* e *padrões concretos de casos de uso (Concrete Use Case Patterns - Cde-casos-de-uso-reificados (Reified Use Case Patterns – RUCP))*, respectivamente. Ambos padrões utilizam *árvores de reificação de IUs*, as quais contêm uma hierarquia de artefatos reusáveis que implementam nos mais variados níveis de detalhe as soluções definidas pelos padrões.

A definição desta hierarquia será apresentada na seção a seguir. Nas seções 5.2 e 5.3 apresentamos em maior detalhe cada um desses padrões. Os processos de reuso na abordagem proposta e a aplicação dos padrões de forma integrada a um processo de desenvolvimento de software dirigido por casos de uso serão apresentados na seção 5.5.

5.1 Árvore de Reificação de IU

A nossa abordagem de reuso foi inspirada no modelo de desenvolvimento de IUs para aplicações interativas multi-contexto, o Cameleon Reference Framework (CRF) [Calvary et. al. 2003], apresentado na seção 4.2.3. O CRF propõe decompor uma IU em artefatos de vários níveis de abstração que se complementam e se relacionam entre si formando uma hierarquia que os organiza no que chamamos de *árvore de reificação de IU*.

Desta forma, o modelo proposto por Calvary, Coutaz, Thevenin, Limbourg, Bouillon e Vanderdonckt se mostra adequado para uma abordagem de reuso de IUs justamente pela possibilidade de organizarmos uma hierarquia de artefatos reusáveis que são identificados e (re)utilizados ao longo de todo o ciclo de desenvolvimento da IU.

Conforme visto na seção 4.1, no projeto da IU, uma das primeiras coisas que o projetista de IU faz é um esboço (ou protótipo) da IU. O protótipo é concebido levando em conta os requisitos documentados nos casos de uso (e que no CRF são representados pelo passo de desenvolvimento Tarefas e Conceitos (T&C)).

Analisando os passos de desenvolvimento do CRF, verifica-se que nenhum destes passos consegue representar adequadamente o protótipo da interface já que uma AUI contém abstrações de *containers* e de conceitos associados, uma CUI é uma especificação formal da interface e uma FUI já é a própria interface implementada em uma linguagem de programação.

Por isso, propomos estender o CRF de forma a introduzir um passo de desenvolvimento que represente o conjunto de artefatos utilizados na prototipação das interfaces com o usuário. Este passo é chamado de *interface com o usuário esboçada* (SUI, do inglês *Sketched User Interface*).

Além de introduzir a SUI no modelo, propomos que a FUI - como sendo o passo de desenvolvimento que representa a implementação da IU - seja desmembrada em dois passos: FUI e XUI, *Final User Interface* e *eXecutable User Interface*, respectivamente.

Uma FUI, conforme a definição original, representa tanto o código-fonte quanto o código-objeto da IU. Como códigos-fonte e códigos-objeto são tipos de artefatos distintos e obtidos por processos de transformação (abstração, reificação e tradução) distintos, consideramos mais adequado separá-los no modelo de forma a explicitar estes níveis de abstração e os mecanismos de transformação necessários para obtê-los. Desta forma, a FUI passa a representar apenas o código-fonte da IU enquanto que o código-objeto (ou executável) da IU fica representado na XUI.

A figura 5.1 contém uma ilustração simplificada com os novos passos que estendem o Cameleon Reference Framework.

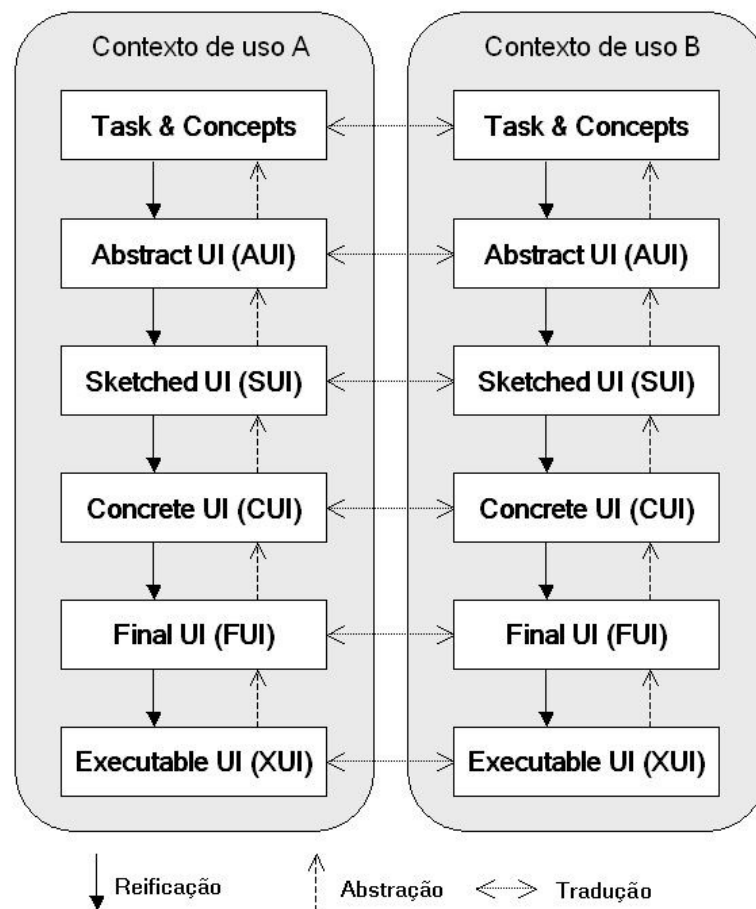


Figura 5.1: O Cameleon Reference Framework Estendido.

As peculiaridades de cada passo de desenvolvimento, na ótica da nossa abordagem, são comentadas a seguir:

- Task & Concepts (T&C): A tarefa a ser executada e os conceitos orientados ao domínio requeridos para o desempenho desta tarefa são obtidos dos padrões de casos de uso e dos padrões de interação.

- Abstract UI (AUI): A AUI representa os mesmos conceitos propostos pelo ~~CRFTask & Concepts (T&C):~~ A tarefa a ser executada e os conceitos orientados ao domínio requeridos para o desempenho desta tarefa são obtidos dos padrões de casos de uso e dos padrões de interação.

- Sketched UI (SUI): As SUIs nada mais são do que um conjunto de artefatos com o esboço (desenho, *screenshot*, etc) e uma descrição textual da IU sendo modelada. Deve existir uma, e somente uma, SUI por modelo de IU. A SUI ocupa papel fundamental na árvore de reificação de IU, pois todos os demais artefatos são formalizações e detalhes da IU descrita na SUI.

A figura 5.2 apresenta um exemplo de SUI para uma IU do padrão de interação “Parts Selector” de van Welie [Welie 2006].

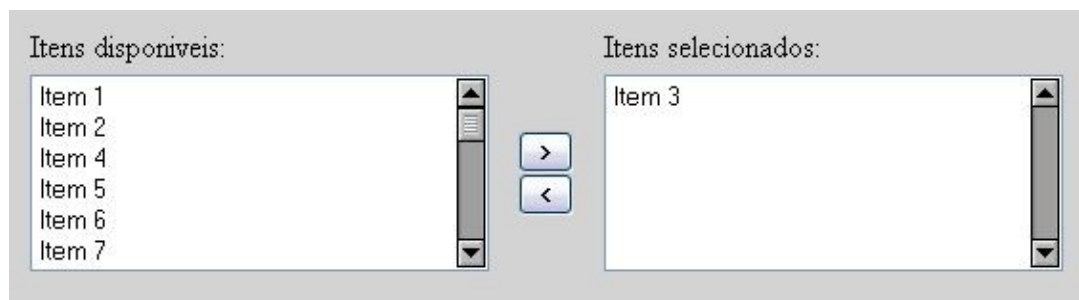


Figura 5.2: Exemplo de SUI .

O único pré-requisito para o projetista construir uma SUI é o conhecimento do *vocabulário* da IU a ser projetada. Desta forma, fica estabelecido quais são os objetos de interação possíveis de serem utilizados bem como o modo de interação.

Como as SUIs são protótipos de IUs, não existe nenhum comprometimento com a tecnologia a ser utilizada nem com aspectos arquiteturais de implementação do software. Se os artefatos da SUI não estão no formato digital e nem é possível digitalizá-los (através, por exemplo, de um scanner ou fotografia digital), deve ser criada uma SUI na árvore de reificação com apenas a referência do local físico onde os artefatos poderão ser encontrados.

- *Concrete UI (CUI)*: Os artefatos que compõem uma CUI são arquivos em XML escritos em uma Linguagem de Descrição de IU (UIDL). Atualmente, as UIDLs que permitem descrever CUIs no maior espectro de ambientes de interação são: UIML, UsiXML e XIML. Neste trabalho, nós optamos por descrever as CUIs em UIML por dois motivos básicos: porque é um dos padrões abertos de UIDL mais utilizados atualmente e porque tem grande flexibilidade para definir quaisquer conjuntos de elementos de interação (*widgets*) através da customização dos seus vocabulários.

Uma CUI pode encapsular (usar) CUIs de outros modelos de IU. Com isso, além da prática de reuso que o encapsulamento proporciona, consegue-se descrever como vários padrões de interação funcionam de forma integrada em num mesmo contexto de IU. Esta prática reforça o desenvolvimento de uma linguagem de padrões na coleção de padrões de interação sendo utilizada.

Para aumentar as possibilidades de reuso de IU e facilitar os processos de abstração, reificação e tradução, sugere-se que as CUIs sejam descritas em n níveis de detalhamento, a saber: CUI básica, CUIs intermediárias e CUI completa. A CUI básica contém apenas a discriminação dos objetos de interação, o layout e descrições textuais de comportamentos relevantes da IU. A CUI completa contém uma especificação detalhada o suficiente para que seja gerada uma FUI a partir dela. Obviamente, as CUIs intermediárias contêm descrições no meio termo entre as duas.

A figura 5.3 apresenta um exemplo de uma CUI escrita em UIML para a interface apresentada na figura 5.2.

```

<?xml version="1.0"?>
<!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 3.0a Draft//EN"
http://uiml.org/dtds/UIML3_0a.dtd>
<uiml>
  <peers> <presentation base="GenericJH_1.2_Harmonia_1.0"/></peers>
  <interface>
    <structure>
      <part class="G:TopContainer" id="top">
        <style> ... </style>
        <part class="G:TextBox" id="textDisp">
          <style>
            <property name="g:textboxtype">textarea</property>
            <property name="g:text">Itens disponiveis:</property>
            <property name="g:editable">>false</property>
          </style>
        </part>
        <part class="G:List" id="listaDisp"> ... </part>
        <part class="G:Button" id="botaoAdiciona" >
          <style>
            <property name="g:text">&gt;</property>
            <property name="g:buttontype">push</property>
            ...
          </style>
        </part>
        <part class="G:Button" id="botaoRemove" > ... </part>
        <part class="G:TextBox" id="textSelec"> ... </part>
        <part class="G:List" id="listaSelec"> ... </part>
      </part>
    </structure>
    <behavior> ... </behavior>
  </interface>
</uiml>

```

Figura 5.3: Extrato de uma CUI para o padrão “Parts Selector”

- *Final UI (FUI)*: É uma IU descrita em uma linguagem de programação, ou seja, são arquivos de texto com os códigos-fonte de linguagens que implementam a IU. Uma FUI pode encapsular (incluir) outras FUIs e invocar chamadas a XUIs.

- *Executable UI (XUI)*: É um programa de computador executável em uma plataforma computacional que renderiza a IU. Geralmente, artefatos reusáveis na forma de XUIs são componentes de IU implementados em tecnologias de objetos como Corba, OCX, JavaBeans, etc.

O CRF Estendido define o modelo de IU como a base sob a qual os artefatos reusáveis serão identificados, produzidos, consumidos e organizados. Assim, o modelo de IU é uma árvore de reificação de IU com artefatos em todos ou em alguns níveis de abstração (amplitude vertical) e com n ramificações em cada nível (amplitude horizontal).

Na UML, o modelo de IU é representado pelo diagrama de classes apresentado na figura 5.4.

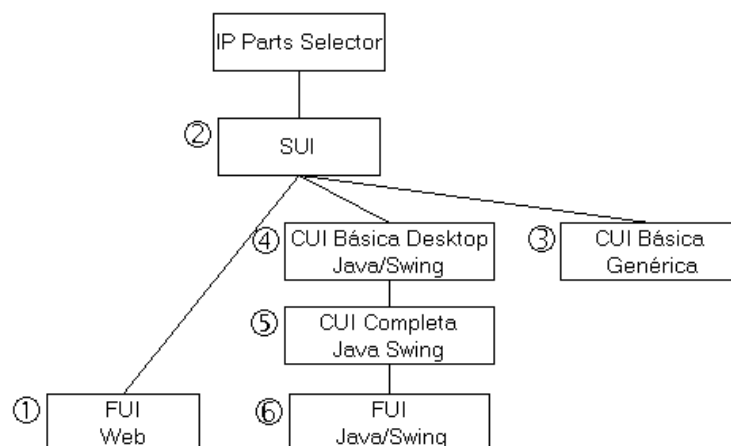


Figura 5.5: Exemplo de árvore de reificação de uma IU.

5.2 Padrão Concreto de Interação (CIP)

Desenvolver interfaces usando um processo de reuso possui basicamente 3 etapas: 1) realizar buscas em algum repositório de elementos reusáveis (*assets*), 2) selecionar os elementos mais adequados e 3) adaptá-los ao novo contexto específico de uso. Hoje, os *assets* de IHC mais comuns presentes em um repositório são os objetos de interação em nível de implementação - e os padrões de interação (*interaction patterns*) - em - num nível de implementação - e os padrões de interação (*interaction patterns*) - num nível mais abstrato.

O objetivo desta seção é apresentar uma abordagem para reuso de interfaces com o usuário (IUs) através da especificação e utilização de modelos de reificação de padrões de interação.

No estágio atual, os padrões de interação têm um potencial muito grande para serem úteis no reuso de projeto de interfaces. No entanto, para isto ocorrer de forma efetiva, algumas questões precisam ser endereçadas. Uma delas se refere à padronização do formato. Embora existam propostas de padronização do formato do padrão (entre elas [Gaffar et al. 2005] e [PLML 2006]), seu objetivo é estruturar a documentação do padrão e não a descrição da IU que implementa a sua solução. Outra questão está relacionada à especificação das interfaces que implementam o padrão: não existem descrições precisas dos elementos de interação e como eles devem se comportar numa implementação da solução proposta pelo padrão de interação. De fato, pode haver diferentes implementações do mesmo padrão com diferenças de comportamento entre elas. Para ilustrar isto, considere o padrão de interação “Parts Selector” da coleção de van Welie [Welie 2006] (reproduzido no Anexo A) a IU que implementa o padrão de interação “Parts Selector” da coleção de van Welie [Welie 2006] apresentada na figura 5.2.

A documentação deste padrão define claramente o trinômio problema-contexto-solução, bem como apresenta exemplos de uso deste padrão. Entretanto, esta documentação não é suficiente para que um projetista de interface possa entender o funcionamento da interface e implementá-la de uma forma consistente com as demais implementações da mesma interface. Muitas questões ficam em aberto. Por exemplo: um item a ser adicionado na caixa de listagem “Itens selecionados” será inserido no final da lista ou no lugar que ocupará em uma lista ordenada? Os itens que são

movimentados da caixa “Itens disponíveis” para a caixa “Itens selecionados” são removidos ou permanecem na caixa “Itens disponíveis”? Podem-se selecionar múltiplos itens ou somente um de cada vez? Como funciona a interface com o uso do teclado? Os botões de movimentação ficam habilitados quando não há item selecionado ou uma caixa de listagem fica vazia?

Para contornar estes problemas, propomos que a documentação do padrão de interação seja estendida por um conjunto de descrições das IUs que o implementam. Em outras palavras: que o padrão de interação seja estendido por uma árvore de reificação de IU. Desta forma, as dúvidas de funcionamento da IU são dirimidas e - o que é mais importante - dá consistência às IUs pelo fato de poderem ser implementadas da mesma maneira.

Com a extensão do padrão de interação - além de endereçar os problemas de consistência e de dubiedade citados acima - vislumbram-se oportunidades de reuso não só das (boas) idéias do padrão de interação, mas também de inúmeros artefatos que vão desde o reuso das descrições de IUs nos projetos de interface até o reuso direto de implementações destas IUs. O conjunto destes artefatos é denominado de padrão concreto de interação, que será definido a seguir.

Um padrão de interação, se for utilizado no projeto de uma IU, na prática, precisa ser implementado por um conjunto de objetos de interação de alguma plataforma, bem como ser codificado em uma linguagem de programação. Obviamente, um mesmo padrão de interação pode ser mapeado para diferentes formas de interação (modalidades) e de implementação (tecnologias). A nossa proposta busca uma abordagem de reuso de IHC ao aglutinar os conceitos de padrão de interação com os de árvore de reificação de IUs no que chamamos de padrão concreto de interação. Desta forma, explicitam-se os elementos de IU necessários para realizar a interação bem como os seus comportamentos, o que torna possível reusar e derivar implementações para as mais diversas plataformas e tecnologias.

Portanto, um padrão concreto de interação (*concrete interaction pattern* – CIP) é um padrão de interação estendido por modelos de IUs que descrevem e implementam as IUs desse padrão de interação. O CIP estende a documentação do padrão de interação com a agregação de árvores de reificação de IUs. Um CIP pode conter inúmeros modelos de IU associados, conforme apresentado na figura 5.56 (o detalhamento da classe UI é apresentado na figura 5.4).

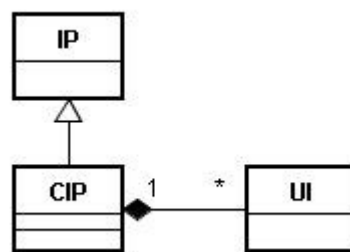


Figura 5.5: Estrutura de um Padrão Concreto de Interação6: Estrutura de um padrão de interação concreto.

Geralmente, um modelo de IU de um CIP é composto por uma SUI, uma AUI e conjuntos de CUIs, FUIs e XUIs que as descrevem em diferentes níveis de detalhamento e para diferentes modalidades de interface (conjunto de *widgets*) e tecnologias de implementação (plataformas/linguagens). UI de um CIP é composto por

uma SUI, uma AUI e conjuntos de CUIs, FUIs e XUIs que as descrevem em diferentes níveis de detalhamento e para diferentes modalidades de interface (conjunto de widgets) e tecnologias de implementação (plataformas/linguagens).

O diagrama da figura 5.6 apresenta um exemplo de uma árvore de reificação de IU para o padrão de interação “Parts Selector”. As figuras 5.2 e 5.3 ilustram, respectivamente, os artefatos 2 e 3 do exemplo. Algumas seqüências possíveis de construção dos artefatos desta árvore seriam: 1-2-3-4-5-6, 2-4-5-6-3-1 e 1-6-2-5-4-3. Tendo como base o padrão PLML [PLML 2006] para a descrição de padrões de interação (ver Anexo A), propomos uma DTD para o CIP. Esta DTD é apresentada no Apêndice A.

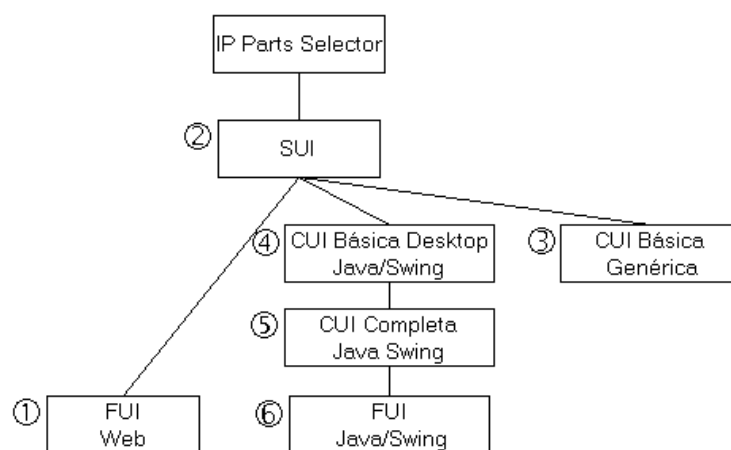


Figura 5.6: Exemplo de árvore de reificação de um CIP.

Tendo como base o padrão PLML [PLML 2006] para a descrição de padrões de interação (ver Anexo B), propomos uma DTD para o CIP. Esta DTD é apresentada no Apêndice A.

5.3 Padrão Concreto de Caso de Uso (CUCP)

5.4 Padrão de Caso de Uso Reificado (RUCP)

Vimos, na seção 3.2.1, que um caso de uso é definido na UML como “uma descrição de um conjunto de seqüências de ações, inclusive variantes, que um sistema executa para produzir um resultado de valor observável por um ator” [OMG 2003].

O metamodelo de caso de uso da UML (apresentado na figura 3.12) não especifica essas “seqüências de ações” ou “interações”. Entretanto, para que possamos aplicar a nossa abordagem de reuso de IHC, é necessário explicitar estas interações através da identificação dos fluxos e passos que compõem a narrativa do caso de uso. Desta forma, propomos que o metamodelo de casos de uso seja estendido para o metamodelo apresentado na figura 5.7.

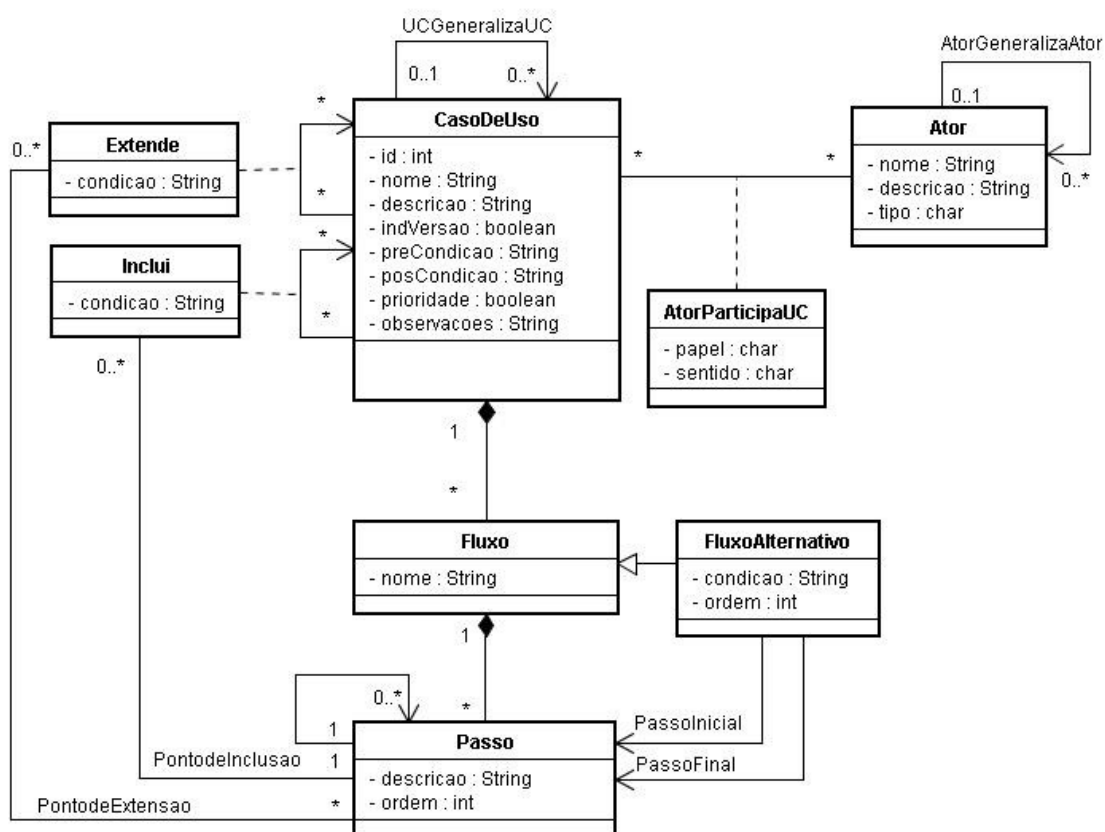


Figura 5.7: Metamodelo de Caso de Uso Estendido

Com o metamodelo de casos de uso estendido é possível se fazer o mapeamento entre cada passo do caso de uso com a IU que suporta a execução desse passo. Isso permite decompor a IU do caso de uso em subconjuntos de espaços de interação e de elementos de interação necessários à execução do passo. Cada passo pode estar associado à pelo menos um objeto de interação em que se dará a interação descrita pelo passo. A figura 5.8 apresenta um diagrama de classes deste mapeamento.

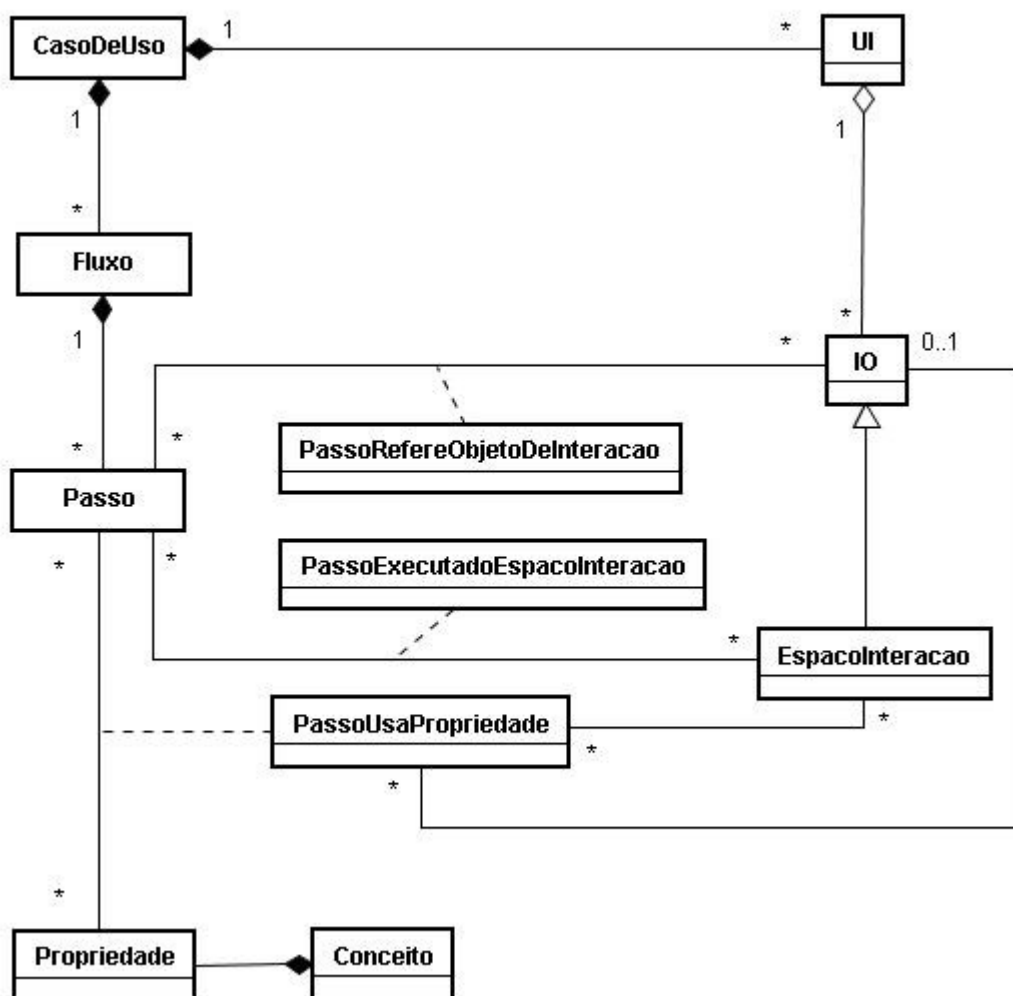


Figura 5.8: Mapeamento entre casos de uso e interfaces com o usuário.

Conforme ~~mo~~demonstrado na figura 5.8, um `CasoDeUso` pode ter vários modelos de UI. Como o modelo de UI é composto de objetos de interação (IOs) (ver seção 4.2.2), pode-se fazer o mapeamento do passo do caso de uso com os espaços de interação e os IOs que suportam o passo (representado no diagrama pelas classes associativas `PassoRefereObjetoDeInteracao` e `PassoExecutadoEspacoInteracao`, respectivamente). Também é possível mapear as propriedades dos conceitos de domínio utilizadas no passo (representadas na classe `PassoUsaPropriedade`) com os espaços de interação e os elementos (ou mecanismos) de interação que as representem na IU.

Na nossa abordagem, o principal elemento de ligação entre o caso de uso a ser desenvolvido e os possíveis artefatos a serem reusados é o padrão de caso de uso.

Da mesma forma que um padrão concreto de interação, um padrão de caso de uso pode conter vários níveis de reificação de IU, formando uma árvore de reificação de IUs que modelam e implementam a IU do padrão de caso de uso. Ao conjunto de artefatos que compõem a hierarquia juntamente com o padrão de caso de uso, chamamos de *Padrão Concreto de Caso de Uso (Concrete Use Case Pattern - Cpadrão de caso de uso reificado (reified use case pattern - RUCP).*

De forma semelhante à estrutura de um CIP, a estrutura de um **CRUCP** é apresentada na figura 5.9.

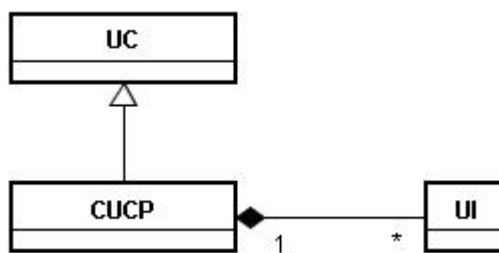


Figura 5.9: Estrutura de um padrão concreto de caso de uso reificado.

No **CRUCP**, uma SUI é o protótipo que constitui a apresentação do padrão de caso de uso. Todos os demais elementos constituem os vários níveis de reificação deste protótipo. Como o padrão de caso de uso deve ser escrito na forma essencial (independente de tecnologia), pode-se associar a este padrão hierarquias de IUs nos mais diversos tipos de ambiente de interação e de tecnologias de implementação. Um único padrão de caso de uso pode, por exemplo, conter SUIs e CUIs que o apresentem em um ambiente de interação Web, em um ambiente gráfico desktop, em um ambiente de um telefone celular, e assim por diante. Para cada um destes ambientes, podem existir FUIs e XUIs que os implementem nas mais variadas linguagens de programação e/ou arquiteturas.

Da mesma forma que se utilizam padrões de interação no projeto da IU de um caso de uso, as IUs de um **CRUCP** também podem ser obtidas com o subsídio de vários CIPs. Isto possibilita um mapeamento entre esses padrões conforme ilustrado na figura 5.10. Com os padrões associados, permite-se - como veremos na seção 5.5 - o uso integrado e sistemático destes padrões no projeto e reuso de IHC.

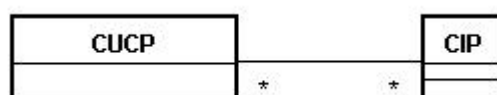


Figura 5.10: Mapeamento entre padrões de casos de uso e padrões de interação

5.5 Padrão de Interface de Gerenciamento do Sistema (SMUIP)

Como vimos na seção 4.1, as IUs de um sistema, através do processo de desenvolvimento dirigido por casos de uso, com o processo de desenvolvimento dirigido por casos de uso são obtidas a partir dos casos de uso. Com isso, seria natural afirmar que o conjunto de IUs do sistema é composto pela soma das IUs de cada caso de uso. Entretanto, não é o que normalmente acontece. Além das IUs dos casos de uso, também é preciso que o projetista de interface defina como organizar todas as IUs do sistema de forma a dar um “corpo” ao sistema como um todo.

Devem ser definidos a forma de integração, navegação e acesso aos vários casos de uso disponibilizados ao usuário. Além disso, possivelmente existirão funcionalidades ortogonais e aplicáveis aos vários casos de uso (tais como as funções de Recortar/Copiar/Colar, Desfazer/Refazer, o uso e customização de barras de ferramentas, a organização de menus e as ações de arrastar-e-soltar (*drag & drop*), para citar algumas) que precisam ser endereçadas no projeto da IU do sistema.

Um leitor mais atento pode alegar que estas funcionalidades nada mais são do que outros casos de uso do sistema e - do ponto de vista da análise de sistemas - de fato são. Entretanto, estes casos de uso raramente são explicitados e incorporados ao conjunto de casos de uso do sistema, talvez por representarem ações do usuário secundárias em relação às demais tarefas do domínio do sistema. Portanto, todo o sistema que utiliza a modelagem de casos de uso sempre possui alguns *casos de uso implícitos*, o que não significa que eles não devam ser construídos.

O conjunto de IUs dos casos de uso implícitos que suportam as funcionalidades de integração e gerenciamento das IUs do sistema é chamado de *Interface de Gerenciamento do Sistema* (*System Management User Interface* - SMUI). É na SMUI que também são definidos o uso ou não de metáforas na IU, bem como questões de estilo (*look & feel*) que influenciarão o projeto das demais IUs do sistema. A principal característica de uma SMUI é a de executar o papel de *integradora* das várias funcionalidades do sistema. A SMUI molda o corpo do sistema, o “palco” onde os usuários executarão as várias tarefas de forma integrada e consistente.

A figura 5.11 apresenta o modelo de uma SMUI e como ela se relaciona com as demais IUs.

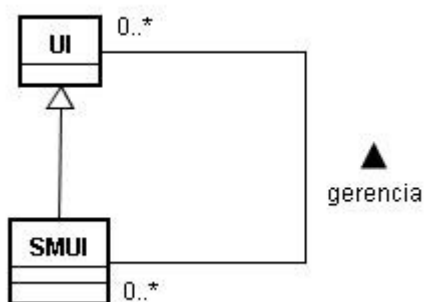


Figura 5.11: System Management User Interface - SMUI.

São exemplos de SMUIs, no ambiente desktop: a interface MDI da suíte MS Office e a interface Explorer-like do MS Explorer; e no ambiente Web: a interface de comércio eletrônico da empresa Amazon. A figura 5.12 contém um *screenshot* de um SMUI Web.



Figura 5.12: Exemplo de SMUI Web

A figura 5.12 ilustra uma típica SMUI utilizada em sites de comércio eletrônico. Neste exemplo, a SMUI define claramente 3 grandes áreas de interação do usuário: (A) área de navegação geral, onde o usuário tem acesso a todos os serviços disponíveis no site; área de navegação por tópicos, subdividida em 2 grupos: (B) navegação por departamento e (C) navegação por categoria de produto; e área de dados (D). A área de dados é o espaço de interação onde são apresentadas as IUs dos principais casos de uso do sistema. Observe que a SMUI define elementos de interação presentes em todas as páginas do site, tais como: acesso à página inicial (*home-page*) (E), o mecanismo de busca por categoria (F) e o acesso ao carrinho de compras (G).

Particularmente, estamos interessados nas SMUIs que as equipes de desenvolvimento constroem para serem reusadas em inúmeros projetos de software. Uma SMUI construída com vista ao reuso constitui um *Padrão de Interface de Gerenciamento do Sistema* (*System Management User Interface Pattern* – SMUIP).

Como um SMUIP é um tipo especial de IU, ele pode ser descrito por uma árvore de reificação de IUs como qualquer IU. Não é raro que, como são IUs muito mais complexas (pois envolvem a organização de inúmeros objetos de interação), sejam disponibilizadas como *frameworks* completos (FUIs e XUIs), fazendo com que o esforço de desenvolvimento do seu uso fique restrito a pequenos ajustes e customizações no código do SMUIP.

Escolher um SMUIP adequado (ou definir uma SMUI) é uma das primeiras decisões que o projetista de IU deve tomar no desenvolvimento do sistema, pois ela estabelecerá um contexto de interação cujas características e restrições de design deverão ser seguidas pelas demais IUs do sistema.

5.6 Processos de reuso na abordagem proposta

As práticas de reuso na ES (revisadas na seção 2.4) empregam o reuso de inúmeros tipos de artefatos que são utilizados em todas as fases do processo de desenvolvimento do software. Pratica-se o reuso desde artefatos mais conceituais (tais como modelos, diagramas e padrões) até os mais concretos (como códigos-fonte e componentes executáveis). A disponibilidade de artefatos reusáveis pode radicalmente melhorar a habilidade de desenvolver software [Kroll 2003].

Uma característica comum em praticamente todas essas abordagens de reuso é quanto ao escopo do reuso restringir-se à atividades específicas do processo de desenvolvimento, ou seja, quando um artefato é reusado, dificilmente esse “reuso” se propaga para os demais artefatos que serão produzidos nas atividades subseqüentes do processo. Por exemplo: na análise de requisitos, tipicamente, reusa-se padrões de casos de uso. A partir do momento em que um padrão de caso de uso é utilizado para descrever um dos casos de uso do sistema, as atividades subseqüentes de implementação do caso de uso seguem normalmente indistintamente do fato de um padrão de caso de uso ter sido ou não utilizado na atividade anterior.

Não há dúvida de que tais práticas têm causado um impacto positivo na forma de desenvolver software. Entretanto, os processos de identificação, produção e consumo destes artefatos reusáveis, geralmente, adotam abordagens que estão dissociadas de um modelo que os represente em vários níveis de abstração. Também não é raro que sejam concebidos sem uma contextualização adequada ao modelo de software como um todo e de compatibilidade restrita a determinados ambientes tecnológicos (o que limita enormemente as oportunidades de reuso).

Quando se fala em reuso de interfaces, a principal questão que tem de ser resolvida é quanto à obtenção e a manutenção da usabilidade desta interface. No momento do projeto da IU, não importa saber como ela vai ser especificada para alguém desenvolver, tampouco com o quê e como ela vai ser construída. Da mesma forma, ao se procurar uma solução de implementação de uma interface, as questões de design e usabilidade já devem ter sido resolvidas, pois resolvê-las simultaneamente pode implicar que determinado aspecto do desenvolvimento prevaleça sobre o outro, ou seja, dependendo do perfil da pessoa ou da equipe de desenvolvimento encarregada de prover a solução, essa solução pode vir com um “viés” de engenharia de software ou da engenharia de usabilidade.

Conforme discutido na seção 4.1, existem esforços de integração das áreas de ES e IHC e isto segue uma tendência atual de mudança de atitude da ES em face de fatores humanos.

A principal característica da nossa abordagem de reuso de IUs é o de conceber artefatos reusáveis em diversos níveis de abstração (que vão desde o nível conceitual até o físico) organizados em hierarquias de forma que se possa trabalhar com o reuso em cada um dos níveis hierárquicos. Organizar artefatos de reuso desta forma permite que se “quebre” as inúmeras questões que têm que ser respondidas para a concepção de um artefato de reuso em questões menores, o que permite o manejo mais adequado do problema como um todo. Esta abordagem é particularmente útil pois permite o envolvimento de equipes multidisciplinares na concepção e construção dos artefatos reusáveis, alocando o profissional com o perfil de conhecimento mais adequado par(skills) mais adequado para cada trabalhar com cada tipo de artefato.

Desta forma, a decisão sobre os artefatos reusáveis que devem ser utilizados e produzidos ao longo de todo o processo de desenvolvimento de um caso de uso é tomada ao longo deste mesmo processo de forma integrada e pontual com relação aos tipos de artefatos que são mais adequados a cada fase do processo. A tabela 5.1 relaciona as macro-atividades de um processo de desenvolvimento com os principais artefatos que são reusados e os papéis de profissionais envolvidos em cada macro-atividade.

Tabela 5.1: Artefatos reusáveis em cada fase do processo de desenvolvimento.

Atividade	Artefato reusável	Papéis envolvidos					
		Analista de Sistemas	Projetista de Interface	Analista de Usabilidade	Arquiteto de Software	Projetista OO	Desenvolvedor
Modelagem de negócios	Padrão de domínio	●					
	Padrão de caso de uso de negócio	●					
Requisitos	Padrão de caso de uso	●					
	Modelo (<i>template</i>) de caso de uso	●					
Análise e Projeto	Arquitetura de sistema				●	○	
	Padrão de projeto				●	●	
	Padrão de interação		●	●			
	AUI	●	○				
	SUI		●				
	CUI		●				○
Implementação	Framework				●	●	●
	Código-fonte					○	●
	Componente				○	○	●
	FUI					○	●
	XUI					○	●

Notação: ● – participação principal. ○ – participação secundária.

Nas subseções seguintes, serão discutidos os processos de consumo, produção, identificação e gerenciamento dos artefatos reusáveis apresentados na nossa abordagem. Para isso, seguiremos a abordagem ABD (*Asset Based Development*), descrita na seção 2.6, por representar um conjunto típico de processos na área de reuso. Entretanto, não é nossa pretensão que a abordagem de reuso de IU aqui apresentada siga estritamente as atividades destes processos, mas o objetivo é ilustrar à luz de uma abordagem já consolidada de processos de reuso, como esses processos funcionariam para a obtenção e consumo dos artefatos de reuso da nossa abordagem.

Atividade	Bem reusável	Papéis
Modelagem de negócios	Padrão de domínio	Analista de Sistemas
	Padrão de caso de uso	
Requisitos	Padrão de caso de uso	Analista de Sistemas

	Padrão de caso de uso reificado	
	Modelo (template) de caso de uso	
Análise e Projeto	Arquitetura de sistema	Analista de Sistemas
	Padrão de projeto	Projetista OO
	Padrão de interação	Projetista de Interface
	Padrão concreto de interação	Designer
	AUI	
	SUI	
	GUI	
Implementação	Framework	Arquiteto de Software
	Código-fonte	Projetista OO
	Componente	Desenvolvedor
	FUI	
	XUI	

~~Nas subseções seguintes, serão discutidos os processos de consumo, produção, identificação e gerenciamento dos artefatos reusáveis apresentados na nossa abordagem. Para isso, seguiremos a abordagem ABD (*Asset Based Development*), descrita na seção 2.5, por representar um conjunto típico de processos na área de reuso. Entretanto, não é nossa pretensão que a abordagem de reuso de IU aqui apresentada siga estritamente as atividades destes processos, mas o objetivo é ilustrar à luz de uma abordagem já consolidada de processos de reuso, como esses processos funcionariam para a obtenção e consumo dos artefatos de reuso da nossa abordagem.~~

5.6.1 Processo de consumo

Um caso de reuso existe quando são encontrados e aplicados artefatos para serem reusados a partir da especificação de um caso de uso.

~~Na nossa abordagem, o principal elemento de ligação entre o caso de uso a ser desenvolvido e os possíveis artefatos a serem reusados é o padrão de caso de uso reificado.~~ A identificação dos possíveis artefatos a serem reusados segue uma orientação *top-down*, ou seja, são analisadas as possibilidades de reuso a partir de especificações mais conceituais dos padrões de casos de uso (níveis mais altos da árvore de reificação) até as mais concretas (níveis mais baixos da árvore de reificação). A quantidade de artefatos reusados dependerá das opções disponíveis ao se percorrer o caminho da árvore de reificação que corresponda à tecnologia-alvo e que seja compatível com a arquitetura do SsD (Sistema sendo Demandado).

O conjunto das atividades do processo de consumo é chamado de *Análise de Reuso*. A análise de reuso ocorre de forma integrada às demais atividades do processo de desenvolvimento OO e em 3 fases distintas, a saber: na fase de modelagem de casos de uso, na fase de apresentação de casos de uso e na fase de realização de casos de uso.

A *análise de reuso de casos de uso* ocorre juntamente com a atividade de análise de requisitos do SsD, ou seja, em paralelo e em sincronia com as atividades de modelagem de casos de uso e modelagem conceitual. A *análise de reuso de interação* ocorre na fase de apresentação de casos de uso paralelamente às atividades de projeto e de prototipação da IU. Finalmente, durante a fase de realização dos casos de uso, pode-se

fazer a *análise de reuso de implementação* juntamente com as atividades de projeto OO e de construção do sistema.

A figura 5.13 apresenta um diagrama com as principais atividades do processo de análise de reuso integrado com o processo de desenvolvimento OO de um sistema. Cada uma destas atividades será detalhada a seguir.

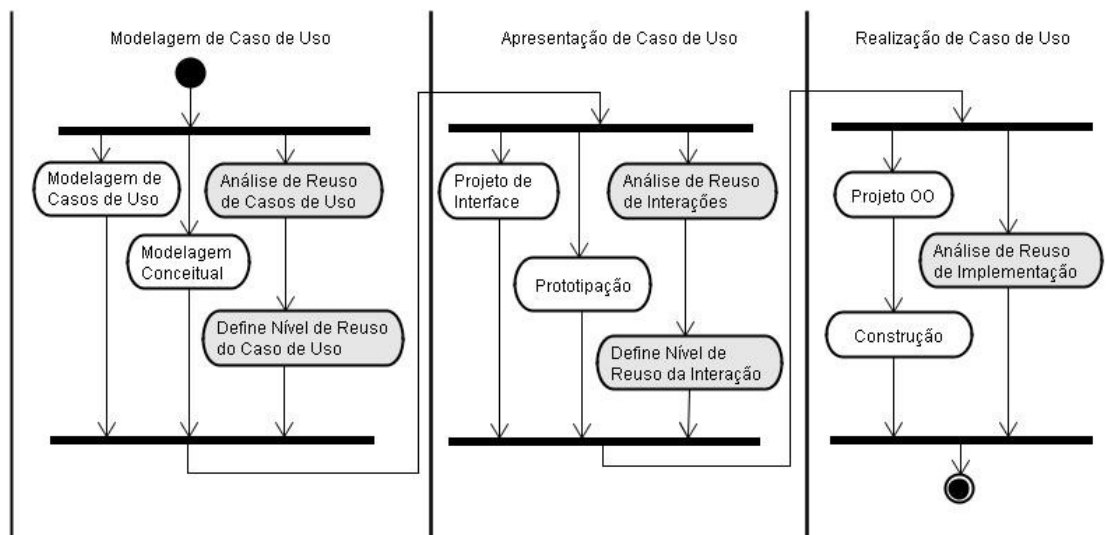


Figura 5.13: Processo de Análise de Reuso integrado ao Processo de Desenvolvimento.

5.6.1.1 Análise de Reuso de Casos de Uso

Na análise de reuso de casos de uso procura-se identificar os padrões de domínio e padrões de casos de uso que sejam aderentes aos requisitos funcionais do SsD. A pesquisa é feita com o enfoque tanto na busca de casos de uso com recorrência de domínio do problema quanto nos casos de uso com recorrência de tarefas padrão. O resultado desta atividade é um conjunto de padrões concretos de casos de uso (Cde-casos de uso reificados (RUCPs)) candidatos a reuso no SsD.

A seguir, para cada CRUCP candidato, procura-se identificar na árvore de reificação, o nível de reuso do caso de uso, ou seja, até que nível de artefatos da árvore de reificação será possível reusar.

Os níveis de reuso de IHC do caso de uso são definidos de acordo com os tipos de artefatos que compõem o Cameleon Framework Estendido. Desta forma, quanto mais reificado for o artefato encontrado para ser reusado, maior será o nível de reuso de IHC do caso de uso. Esses níveis de reuso são apresentados na tabela 5.2.

Tabela 5.2: Níveis de reuso de IHC de um caso de uso.

Fase	Número	Nível de reuso	Comentário
------	--------	----------------	------------

Análise	0	Nenhum	Não há nenhum artefato que possa ser reusado.
	1	Objetivo	Somente o nome do caso de uso e, eventualmente, seus relacionamentos são descritos. Não há a descrição do fluxo de eventos (narrativa).
	2	Narrativa inicial	Contém o fluxo principal do caso de uso descrito, mas ainda não são detalhados os fluxos alternativos.
	3	Narrativa completa	O caso de uso está completamente descrito.
	4	AUI	Os conceitos que participam do caso de uso e os espaços abstratos de interação estão definidos
	5	SUI	Protótipo da interface
Projeto	6	CUI Básica	A CUI Básica serve como especificação inicial da IU. Pode-se definir IUs em ambientes de tecnologia de interação genéricos.
	7	CUI Intermediária	CUIs intermediárias permitem poupar parte do trabalho de especificação de IU.
	8	CUI Completa	A especificação da IU está completa. O reuso neste nível significa que não há necessidade de esforço de análise e projeto para a obtenção da IU deste caso de uso.
Construção	9	FUI	O código-fonte da IU na tecnologia alvo pode ser reusado. O esforço de desenvolvimento neste nível de reuso restringe-se a pequenos ajustes (customizações) no código.
	10	XUI	O reuso em nível de XUI elimina quaisquer esforços de desenvolvimento desta IU (portanto, com 100% de reuso) na medida em que é necessário apenas definir como executar a XUI no contexto do sistema.

Obviamente que a obtenção do nível máximo de reuso de um caso de uso, já nesta fase, está vinculada à definição ou não das tecnologias de interação e de implementação do SsD. Se a tecnologia de interação ainda não está definida, então o nível máximo de reuso será o de AUI (nível 4). Com a tecnologia de interação definida e a de implementação ainda não definida, o nível máximo de reuso passa a ser o de CUI Básica (nível 6). Com ambas as tecnologias definidas, o nível máximo de reuso pode chegar a XUI (nível 10).

A atividade final da análise de reuso de casos de uso consiste em consolidar os casos de uso reusados com os demais casos de uso do SsD (os casos de uso os quais não foi possível identificar nenhum nível de reuso). O resultado pode ser apresentado numa lista do tipo ator-objetivo (ver [Cockburn 2005]) onde acrescentamos duas colunas para registrar o **CRUCP** utilizado e o nível de reuso de IHC.

Para ilustrar o processo de análise de reuso de casos de uso, vamos tomar como exemplo a demanda por um sistema de comércio eletrônico para uma loja de animais de estimação (*pet shop*). Neste exemplo, o cliente necessita de um sistema na Web que suporte todo o processo de comercialização de animais de estimação (cães, gatos e peixes), bem como a toda a linha de produtos relacionados a esses animais. O sistema também deve permitir que os clientes da loja possam agendar os serviços de banho, de tosa e de passeio de cães.

Deve haver um módulo de administração do sistema onde serão mantidas todas as informações para o funcionamento da loja virtual (tais como o cadastro da linha de produtos, dos animais e das pessoas autorizadas a operar o sistema). Este módulo deverá ser desenvolvido como uma aplicação *desktop*. As tecnologias de interação e de implementação já haviam sido definidas na ocasião da análise de reuso de casos de uso deste sistema.

Uma lista de ator-objetivo parcial deste sistema é apresentada na tabela 5.3.

Tabela 5.3: Exemplo de lista ator-objetivo de um sistema de comércio eletrônico.

Nro	Ator	Objetivo ao nível de tarefa	Prioridade	CUCP	Nível de Reuso
1	Cliente	Pesquisa produto	1	Pesquisa {objeto}	5 - SUI
2		Consulta produto	1	Consulta {objeto}	5 - SUI
11		Coloca produto no carrinho	1	Coloca produto no carrinho	5 - SUI
24		Efetua compra	1	Efetua compra	5 - SUI
35		Agenda serviço	3	Agenda {evento}	3 - Narrativa completa
36		Compara produto	4		-
41	Administrador	Inclui produto	1	Inclui {objeto}	9 - FUI
42		Exclui produto	1	Exclui {objeto}	9 - FUI
48		Inclui categoria	1	Inclui {objeto}	9 - FUI
72		Inclui condutor de cães	3	Inclui {objeto}	9 - FUI
76		Inclui serviço	2		6 - CUI Básica
78		Inclui usuário do sistema	1	Inclui usuário	10 - XUI
79		Associa permissões aos usuários	1	Associa permissões aos usuários	10 - XUI
80	Condutor	Informa disponibilidade de horário	3		2 - Narrativa inicial
82		Aceita compromisso de passeio	3		2 - Narrativa inicial
87	Criador	Informa ninhada	1		-
...					
102	Atendente	Registra serviço realizado	2		-

Na análise de reuso de casos de uso desse sistema, constatou-se que havia um padrão de domínio de comércio eletrônico onde foi possível reusar um conjunto de casos de uso que dão suporte ao processo de comercialização de produtos. Os casos de uso de cadastramento de usuários (78) e de permissões de acesso (79) vão ser reusados em nível de XUI devido à utilização de um serviço já existente. O caso de uso referente ao agendamento dos serviços de banho, tosa e passeio (35) pode ter a sua narrativa reusada a partir da identificação de um padrão de caso de uso com tarefa semelhante. Na tabela 5.3, os casos de uso 41, 42, 48 e 72 são casos de uso que serão implementados em uma aplicação *desktop* e, neste ambiente, foi possível identificar um nível de reuso mais alto. Para os casos de uso 36, 87 e 102, nesta análise de reuso não foi encontrado~~Os casos de uso 36, 87 e 102, infelizmente esta análise de reuso não encontrou~~ nenhum artefato que pudesse ser reusado.

As possibilidades de reuso não se encerram com esta atividade. É possível que mais adiante - na fase de apresentação do caso de uso - novos artefatos reusáveis sejam identificados, como veremos a seguir.

5.6.1.2 Análise de Reuso de Interações

Durante a fase de apresentação dos casos de uso ocorre a segunda análise de reuso. Nesta fase, o foco é explorar as possibilidades de reuso de interações para todos os

casos de uso do SsD em que não foi possível identificar, no mínimo, reuso em nível de SUI na análise de reuso anterior.

Esta análise deve ser feita após ter sido definida a tecnologia de interação onde o caso de uso será executado. Em função disso, se as tecnologias de interação e/ou de implementação não haviam sido definidas na ocasião da análise de reuso de casos de uso, então este é o momento de fazer uma revisão do nível de reuso de IU dos casos de uso.

Na fase de apresentação de um caso de uso, uma das primeiras tarefas que o projetista de interface realiza é a de definição de uma SMUI. Essa definição é importante pois, de certa forma, a SMUI impõe restrições quanto ao estilo e o comportamento das IUs que ela gerencia. Conseqüentemente, a busca e seleção dos próximos artefatos reusáveis de IU deve levar em conta esse critério de compatibilidade com a SMUI. Isto posto, pode-se dizer que a busca e seleção no repositório de artefatos reusáveis de um *padrão de interface de gerenciamento de sistema (SMUIP)* inaugura a primeira possibilidade de reuso desta fase.

Conforme ilustrado na figura 5.134, a análise de reuso de interações ocorre simultaneamente com as atividades de projeto e de prototipação da IU. Na medida que o projetista de interface faz a leitura da narrativa do caso de uso e tenta esboçar uma IU que o suporte, também deve fazer uma pesquisa no repositório em busca de padrões concretos de interação (CIPs) que sejam adequados à execução das tarefas descritas nos passos do caso de uso. A busca no repositório deve ser feita de forma recursiva e incremental ao longo do projeto e prototipação da IU e deve levar em conta: a natureza da tarefa, os requisitos de usabilidade e as informações necessárias para um passo ser executado.

Para cada CIP identificado como adequado para a IU que está sendo projetada verifica-se, na árvore de reificação deste CIP, até que nível e quais os artefatos que poderão ser reusados. O critério de escolha destes artefatos deve se basear na existência de um ramo na hierarquia do CIP que seja aderente às tecnologias de interação e de implementação da IU, e compatível com o SMUI e a arquitetura do SsD.

Importante salientar que um CIP, para ser reusado, não necessariamente precisa dar suporte para a apresentação de todos os passos do caso de uso. É comum acontecer que um padrão de interação sirva apenas para suportar uma interação específica dentro de um passo do caso de uso, ficando as demais interações para serem definidas e projetadas pelo projetista de interface.

O resultado da análise de reuso de interações deve ser o de identificar e incorporar artefatos reusáveis da árvore de reificação dos CIPs através do mapeamento com as IUs projetadas para suportar os passos do caso de uso sendo desenvolvido.

5.6.1.3 Análise de Reuso de Implementação

A análise de reuso de implementação deve ser feita em paralelo às atividades de projeto OO e de construção do SsD. A esta altura do processo, a tecnologia de implementação deve ter sido definida. Se ela havia sido **definida** até a presente fase, então faz-se necessário uma nova revisão do nível de reuso de IU dos casos de uso e dos artefatos dos CIPs associados às IUs dos casos de uso à luz desta definição tecnológica.

A análise de reuso de implementação deve ser feita para cada artefato de apresentação encontrado. O objetivo é verificar se existe uma implementação deste

artefato a ser reusada que atenda e seja compatível com a tecnologia de implementação do SsD. Isso significa identificar artefatos implementados na mesma linguagem de programação e/ou arquitetura de componentes (DLL, COM/DCOM, J2EE, etc) escolhidos para a construção do SsD. Além disso, a forma como estes artefatos foram construídos deve ser compatível com os aspectos arquiteturais (*frameworks*, padrões de projeto utilizados, etc) do projeto do SsD.

O resultado da análise de reuso de implementação é um conjunto de programas, pedaços de código, FUIs, XUIs, executáveis, componentes e serviços que serão reusados na construção da IU dos casos de uso do SsD.

5.6.2 Processo de identificação

A identificação dos artefatos reusáveis ocorre em três eventos distintos: 1) na ocasião do ~~inventário inicial~~ ~~carga inicial do repositório~~; 2) durante o processo de desenvolvimento de um sistema; e 3) quando ocorre alguma mudança no ambiente tecnológico e/ou de negócio dos sistemas da organização.

5.6.2.1 Identificação através de inventário inicial

Para a carga inicial do repositório, faz-se um inventário e prospecção dos padrões de domínio, de casos de uso e de interação que interessam à organização. Esta tarefa pode ser feita tanto internamente com os sistemas da organização, quanto externamente através da obtenção de coleções de padrões de terceiros (muitas delas disponíveis gratuitamente na Internet). A pessoa ou equipe responsável por esta tarefa faz a triagem baseada nos tipos de sistemas que a organização está desenvolvendo ou pretende desenvolver. A ênfase não deve ser na quantidade de artefatos encontrados, mas sim na qualidade dos artefatos e na sua aderência ao modelo de negócios da organização. Não será muito útil identificar dezenas (ou até mesmo centenas) de padrões para povoar um repositório se eles não tem nenhuma aderência e aplicabilidade direta nos sistemas da organização.

Após o inventário dos padrões já existentes, faz-se um inventário de *artefatos candidatos a reuso*. Neste inventário, procura-se identificar não só padrões de IU “implícitos” como também quaisquer artefatos (modelos, códigos-fonte, executáveis, etc) que têm grande potencial para fazerem parte dos padrões ~~concretos~~ ~~(C)~~ ~~reificados~~ ~~(R)~~ ~~U~~ ~~C~~ ~~P~~s e CIPs), mas que ainda não foram formatados, documentados e catalogados para este fim.

Também é nesta atividade que são identificadas as SMUIs candidatas a se tornar SMUIPs.

O resultado deste inventário deve ser uma coleção de padrões e de artefatos candidatos a comporem o repositório de artefatos reusáveis da organização.

A seguir deve-se fazer, para cada artefato candidato identificado, um plano de constituição da árvore de reificação sob a qual este artefato presumivelmente fará parte. Este plano permitirá identificar novos artefatos candidatos através da opção pela reificação, abstração ou tradução do artefato candidato original. Obviamente que as opções para a constituição desta árvore surgirão de forma a corroborar com o reuso no ambiente tecnológico e de negócios da organização.

Como exemplo, suponha que um código-fonte que implementa uma solução de IU fosse identificado por ser sempre reusado pelos desenvolvedores nos novos projetos. Ao

fazer o plano de constituição da árvore de reificação a partir deste código-fonte, foi possível identificar um CUCP cujos artefatos serão obtidos fundamentalmente por abstração (engenharia reversa) do código-fonte. Este CRUCP cujos artefatos serão obtidos fundamentalmente por abstração (engenharia reversa) do código-fonte. Este RUCP inicialmente terá uma árvore de reificação composta pelos seguintes artefatos: um padrão de caso de uso, uma SUI, uma CUI Básica e uma FUI (oriunda do código-fonte original).

5.6.2.2 Identificação durante o processo de desenvolvimento de um sistema

Ao longo do processo de desenvolvimento de um sistema é possível identificar, tanto pela equipe de projeto quanto pela equipe que dá suporte ao reuso, que determinado artefato produzido para este sistema tem potencial para se tornar um artefato reusável. Para isso, a equipe o indica como artefato candidato a reuso para que ele, posteriormente, seja avaliado no seu real potencial de reuso e, caso esse potencial se confirme, seja elaborado um plano de constituição da sua árvore de reificação.

Várias demandas para a obtenção de novos artefatos e/ou modificação de artefatos existentes podem surgir ao longo do processo de desenvolvimento de um software, dentre elas, podemos citar:

- indicação de candidato a padrão de caso de uso,
- indicação de candidato a padrão de domínio,
- solicitação para a reificação de um padrão de caso de uso,
- solicitação de manutenção corretiva de um artefato,
- solicitação de nova versão de um artefato,
- indicação de candidato a padrão de interação,
- indicação de candidato a SMUIP,
- indicação de nova IU para a apresentação de caso de uso,
- indicação de reificação de um padrão de interação,
- indicação de reificação de um SMUIP,
- indicação de abstração/reificação/tradução de uma IU.

O resultado destas demandas é uma lista de novos artefatos candidatos identificados e de alteração de artefatos reusáveis já existentes no repositório.

5.6.2.3 Identificação por mudanças no ambiente tecnológico ou de negócios

Sempre que uma nova tecnologia ou versão de tecnologia para o desenvolvimento de sistemas for introduzida (seja ela uma nova linguagem de programação, uma arquitetura ou um *framework*) ou uma nova linha de sistemas em outra área de negócio for iniciada, cabe fazer uma revisão de todo o acervo no repositório de forma a identificar novos artefatos reusáveis candidatos a serem reusados com essa nova tecnologia e/ou linha de negócio.

A indicação de novos artefatos reusáveis provocada por uma mudança tecnológica implica, tipicamente, em processos de tradução de artefatos já existentes na árvore de reificação em outros artefatos na nova tecnologia. Por exemplo: se uma organização

contém um repositório de **CRUCPs** e CIPs com FUIs eminentemente escritas em Visual Basic 6.0, ao decidir migrar o seu desenvolvimento para a plataforma Java/Swing, esta decisão implica na indicação de novas FUIs candidatas a serem escritas em Java/Swing, provocando, assim, um crescimento horizontal em todas as árvores de reificação envolvidas.

Já a indicação de novos artefatos reusáveis provocada por mudanças de negócio implica em prospectar novos padrões de casos de uso e de domínio aderentes ao novo ambiente de negócio com conseqüentes desdobramentos oriundos da identificação destes novos padrões.

5.6.3 Processo produção

Todos os artefatos de IU reusáveis apresentados neste trabalho podem ser produzidos e catalogados como *assets* RAS [OMG 2005]. A figura 5.14 apresenta um diagrama de classes que define esses vários artefatos como classes que herdam e estendem as propriedades e métodos da classe *Asset*.

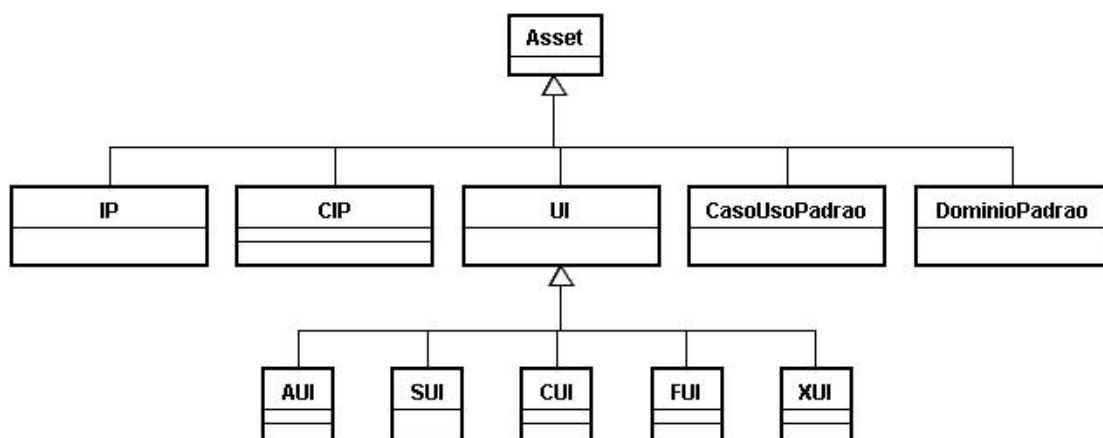


Figura 5.14: Artefatos de IU reusáveis como *assets* RAS

Como vimos na seção 2.4.7, a especificação RAS define 3 perfis de *assets*: *Default*, *Default Component* e *Default Web Service*. Os *assets* de IU da nossa abordagem inicialmente devem ser documentados utilizando o perfil *Default*. Futuramente, para aperfeiçoar e melhorar essa documentação, pretende-se definir e criar dois novos perfis, a saber: *Pattern Profile* e *UI Profile*. O *Pattern Profile* servirá para descrever os diversos tipos de *assets* caracterizados como padrões (padrões de casos de uso, padrões de interação, padrões de domínio, etc) e o *UI Profile* servirá para documentar informações mais detalhadas dos *assets* de IU.

A produção dos artefatos é pautada pelos artefatos candidatos identificados no processo de identificação. A equipe de produção faz uma análise de viabilidade de produção destes artefatos candidatos e decide pela sua produção ou não, quando e como irá produzi-los.

A equipe responsável pela produção dos artefatos reusáveis de IHC deve se ocupar não só com a concepção, desenvolvimento, documentação, empacotamento e disponibilização de cada artefato individualmente, mas também pela integridade e compatibilidade entre os artefatos que compõem uma árvore de reificação. Também deve considerar que a produção de artefatos de IU dependentes de tecnologia de implementação (tais como as CUIs completas, FUIs e XUIs) deve manter a

compatibilidade e a aderência desses artefatos com as arquiteturas, *frameworks* e padrões de programação utilizados nos projetos da organização, bem como a integração e compatibilidade desses artefatos com os demais artefatos não-IU (regras de negócio, *design patterns*, componentes, serviços, etc) presentes no repositório.

Geralmente, a produção de um artefato que irá fazer parte de uma árvore de reificação de um CIP ou **CRUCP** seguirá um dos processos de transformação previstos no Cameleon Framework Estendido (abstração, reificação ou tradução). Estes processos podem ser executados manualmente ou serem automatizados. Os processos automatizados podem realizar transformações segundo a abordagem de *desenvolvimento orientado a modelos* (*model driven architecture - MDA*). Na MDA, os modelos abstratos são chamados de PIMs (*Platform Independent Models*), enquanto que as implementações são chamadas de PSMs (*Platform Specific Models*) [Kleppe et al. 2004].

Na nossa abordagem, uma CUI escrita em um vocabulário genérico é um PIM que poderá ser transformado em PSMs (CUIs de tecnologias específicas ou FUIs). Inversamente, uma XUI ou FUI (PSM) poderá ser transformada automaticamente em CUI, AUI ou SUI (PIM) dependendo apenas da execução adequada do mecanismo de transformação com a definição de transformação previstos na abordagem MDA.

Visto sobre esta perspectiva, pode-se dizer que as transformações automatizadas do Cameleon *Framework* são *instâncias* de processos de transformação da MDA. Ambas abordagens são complementares e perfeitamente compatíveis entre si.

5.6.4 Processo de gerenciamento

As atividades de gerenciamento de *assets* de IHC não diferem significativamente das que são feitas para gerenciar o ciclo de vida de quaisquer outros *assets*. No entanto, algumas observações são necessárias.

O fato de se propor estruturar artefatos reusáveis de IHC em árvores de reificação não implica que a produção destes artefatos deva *obrigatoriamente* constituir árvores completas, ou seja, árvores com pelo menos um artefato em todos os níveis de reificação. Na nossa abordagem, não há imposição sobre quantidade, ordem de obtenção e quais devem ser feitos primeiro.

De fato, a decisão sobre o que deve ou não compor a árvore de reificação deve se basear em critérios estratégicos de gerenciamento de reuso da organização. Sendo assim, a árvore crescerá, ao longo do tempo, de acordo com as necessidades de reuso da organização e somente pro lado que interessa.

Entretanto, cabe ressaltar que árvores de reificação de baixa amplitude vertical (~~taias~~ como CIPs compostos apenas de um padrão de interação e algumas FUIs) diminuem as possibilidades de reuso, pois nestas árvores os elementos de IU em níveis mais abstratos não estão representados adequadamente e é sabido que aumentam as possibilidades de se obter novos artefatos reusáveis à medida que os conceitos envolvidos se tornam mais abstratos.

A questão dos direitos autorais: Tipicamente, CIPs e **CUCPs** contêm múltiplos autores, pois cada elemento na árvore de reificação pode ser criado por um autor diferente. Nada impede que alguém (que não o autor) acrescente um novo elemento de IU em uma árvore de reificação já existente. A autoria, portanto, está em nível de artefato e não do CIP e CRUCPs ~~contêm múltiplos autores, pois cada elemento na~~

~~árvore de reificação pode ser criado por um autor diferente. Nada impede que alguém (que não o autor) acrescente um novo elemento de IU em uma árvore de reificação já existente. A autoria, portanto, está em nível de artefato e não do CIP e RUCP~~ como um todo. Entretanto, se esse alguém necessitar modificar um artefato ao qual não é o autor, então deve tomar os cuidados necessários de forma a não infringir os direitos autorais desse autor. A liberdade que um não-autor terá para modificar um artefato já existente vai depender do tipo de licenciamento de uso deste artefato. Se for uma licença proprietária, então ele terá que respeitar as restrições impostas pela licença que, não raro, pode proibir que sejam feitas quaisquer modificações. Se for uma licença livre (tais como a GNU *General Public Licence* - GPL [GPL 1991] e Creative Commons [CC 2006]), as restrições impostas à modificação são quase inexistentes, dando ampla liberdade para modificar e reutilizar o artefato. Uma das poucas restrições impostas pelas licenças livres está relacionada com a preservação da autoria original e com as liberdades de uso dos artefatos derivados (que geralmente, não poderão ser mais restritivas que a liberdade de uso do artefato original).

Cabe às pessoas responsáveis pelo gerenciamento do repositório de artefatos reusáveis decidir sobre estas questões de licenciamento à qual os artefatos do repositório estarão submetidos.

Isto posto, acredita-se que a adoção de um modelo de desenvolvimento colaborativo pode render grandes benefícios através do esforço coletivo de desenvolvimento/teste/feedback das árvores de reificação, ou seja, a disponibilização e/ou o uso pela organização de um repositório que tenha acesso público e de livre manutenção (disponibilizado na Internet, por exemplo) potencializaria a obtenção de mais artefatos reusáveis, bem como diluiria os custos de produção e documentação destes artefatos entre os vários agentes da comunidade.

5.7 Considerações sobre ferramentas de suporte à abordagem

Assim como em quaisquer outras tarefas realizadas no processo de desenvolvimento de software, a utilização de ferramentas de engenharia de software auxiliadas por computador (CASE) contribui – e muito – para a produtividade e boa execução destas tarefas. Em certos casos a falta de uma ferramenta CASE inviabiliza totalmente a própria execução da tarefa. Por isso, consideramos que a nossa abordagem, para ser empregada na sua plenitude, deve ser suportada por ferramentas CASE.

A especificação destas ferramentas não está no escopo deste trabalho. Entretanto, o tipo de ferramentas que é necessário e as tarefas da nossa abordagem que elas suportariam são comentadas a seguir.

5.7.1 Modelagem de casos de uso e demais modelos da UML

A nossa abordagem faz uso de quaisquer ferramentas de modelagem UML disponíveis no mercado. Entretanto, seria interessante se essa ferramenta implementasse a descrição de fluxos e passos dos casos de uso na forma proposta pelo metamodelo de casos de uso estendido. Por ser necessária a troca de modelos da UML com outras ferramentas, faz-se necessário que esta ferramenta implemente um padrão aberto para intercâmbio de modelos (via XMI).

Também, a ferramenta deveria implementar o suporte à tarefa de mapeamento dos casos de uso e passos de casos de uso com as respectivas IUs que os suportam.

5.7.2 Modelagem de IU

Ambientes de desenvolvimento integrados (*IDEs*) que dão suporte a programação visual são fundamentais para um bom desempenho no trabalho de projeto e construção de IUs.

A nossa abordagem pode fazer uso das principais *IDEs* disponíveis no mercado. Entretanto, cabe ressaltar que estas ferramentas produzem IUs para plataformas tecnológicas e linguagens de programação específicas, ou seja, produzem FUIs. Com isso, a idéia de produzir especificações de IU abertas e independentes de plataforma (propiciado pelas CUIs) fica comprometida.

Para suportar o desenvolvimento de CUIs, as *IDEs* devem gerar, ao invés de código em uma linguagem de programação específica, especificações em XML da IU, ou seja, em uma UIDL tal como a UIML ou UsiXML.

Atualmente já existem algumas ferramentas disponíveis que renderizam IUs a partir de uma especificação em uma UIDL e vice-versa. Brevemente, estas ferramentas atingirão um nível de maturidade semelhante ao encontrado nas melhores IDEs do mercado.

5.7.3 Repositório de *assets*

Tanto os **CRUCPs** quanto os CIPs necessitam de uma ferramenta que permita pesquisar, catalogar, documentar, manter, relacionar e publicar os diversos *assets* que compõem a hierarquia de reificação destes padrões. Repositórios RAS são adequados a este propósito, permitindo, inclusive que se disponibilizem repositórios distribuídos.

A documentação destes *assets* poderá ser melhorada se forem definidos *RAS Profiles* específicos para os tipos de assets de IU da nossa abordagem.

5.7.4 Ferramentas de transformação

Como vimos na seção 5.5.3, os artefatos que constituem as árvores de reificação dos CUCPs e CIPs são modelos aderentes aos modelos PIM e PSM da abordagem MDA [Kleppe et al. 2004]. ~~hierarquias de reificação dos RUCPs e CIPs são modelos compatíveis com os modelos PIM e PSM da abordagem MDA [Kleppe et al. 2004].~~

Desta forma, é possível utilizar as ferramentas de transformação do MDA para criar vários artefatos da árvore, em especial, para executar é possível empregar ferramentas de transformação automática entre os vários artefatos da hierarquia, em especial, nas transformações entre os níveis de CUI, FUI e XUI.

Estas ferramentas de transformação podem implementar um modelo de transformação de IU semelhante ao modelo proposto na especificação da linguagem UsiXML. Mais detalhes sobre estas transformações podem ser obtidos em [UsiXML 2006].

6 EXEMPLO DE APLICAÇÃO

O objetivo deste capítulo é apresentar um exemplo de aplicação da abordagem proposta. O planejamento do trabalho, as características do ambiente e o escopo no qual o estudo de caso foi realizado são apresentados na subseção a seguir. O inventário dos artefatos reusáveis obtidos no processo de identificação e uma breve descrição do sistema e uma breve descrição do sistema a ser desenvolvido, cujo projeto serviu de base para a aplicação do processo de consumo de reuso, são apresentados nas seções 6.2 e 6.3, respectivamente. A seção 6.4 mostra o resultado do processo de reuso dirigido por casos de uso com a apresentação dos artefatos que efetivamente foram identificados e utilizados neste projeto. Finalmente, a seção 6.5 apresenta uma análise e discussão dos resultados obtidos com a aplicação da abordagem.

6.1 Planejamento do estudo de caso

6.1.1 Objetivo

O estudo de caso apresentado a seguir tem como objetivo avaliar e validar a aplicação dos processos de identificação e consumo de artefatos reusáveis que caracterizam o reuso dirigido por casos de uso da nossa abordagem, bem como avaliar e mensurar - quando possível - os ganhos e benefícios obtidos com essa prática de reuso.

Por não diferenciarem significativamente de outras abordagens (ex: [Ambler et al. 2005] e [Larsen & Wilber 2005]) - e por restrições de tempo e de recursos - os processos de produção e gerenciamento de artefatos reusáveis não foram explorados e executados neste estudo de caso.

6.1.2 Características da empresa onde foi executado o estudo de caso

Este estudo de caso foi aplicado num dos projetos de desenvolvimento de software da PROCERGS - Companhia de Processamento de Dados do Estado do Rio Grande do Sul. As informações sobre a PROCERGS apresentadas a seguir foram obtidas diretamente do site da empresa em [PROCERGS 2006].

6.1.2.1A Empresa

A Companhia de Processamento de Dados do Estado do Rio Grande do Sul - PROCERGS, é uma empresa de economia mista, que iniciou suas atividades em 28 de Dezembro de 1972 como órgão executor da política de informática do Estado.

Hoje a PROCERGS é a maior empresa de informática do Rio Grande do Sul e processa diariamente milhões de transações vitais para o bom funcionamento do serviço público e o atendimento à comunidade, afetando a vida de milhões de gaúchos.

Tem como missão atuar como instrumento de vanguarda para a modernização e eficiência do serviço público através da tecnologia da informação e comunicações, em benefício do cidadão.

A PROCERGS dispõe de um acervo de mais de 200 sistemas aplicativos nas mais diversas áreas de atuação do Estado. Com tecnologia avançada e profissionais especializados, a PROCERGS está capacitada a desenvolver e manter sistemas integrados de grande complexidade, ou de missão crítica, com um alto grau de qualidade e confiabilidade.

Além dos órgãos da administração direta e indireta do Estado do Rio Grande do Sul, a PROCERGS também gera soluções para os órgãos dos Poderes Legislativo, Judiciário, e de outras esferas de Governo.

6.1.2.2 Organograma

A PROCERGS está estruturada em três grandes áreas: Diretoria Comercial, Diretoria de Operações e Diretoria de Desenvolvimento.

A Diretoria Comercial engloba as funções relacionadas à administração interna (a administração interna – gestão de recursos humanos, gestão contábil-financeira, gestão administrativa) e comercial (marketing, negócios e coordenadorias regionais).

A Diretoria de Operações engloba as funções de produtos operacionais (estruturada em setores organizados segundo as principais linhas de serviços da empresa: produção, telecomunicações, redes e equipamentos), atendimento/suporte a usuários e tecnologia de operações.

A Diretoria de Desenvolvimento engloba as funções de administração de sistemas (estruturada em setores organizados segundo as áreas de atuação do Estado, visando assegurar a especialização nessas áreas), desenvolvimento de projetos e tecnologia de desenvolvimento.

Atualmente, a Diretoria de Desenvolvimento é composta por 4 Divisões, a saber: Divisão de Administração de Sistemas, Divisão Financeira e Tributária, Divisão de Projetos e Divisão de Tecnologia e Infra-estrutura. Existem mais de 300 profissionais de TI lotados nesta Diretoria.

6.1.2.3 Ambiente de desenvolvimento

Basicamente existem 3 grandes grupos de sistemas sendo desenvolvidos pela PROCERGS: sistemas que rodam em *mainframes*, sistemas de aplicativos desktop e sistemas Web. Os sistemas de *mainframe* são desenvolvidos com as tecnologias Natural/Adabas na plataforma IBM e Cobol na plataforma Unisys. Aplicativos desktop são desenvolvidos com as tecnologias Visual Basic e Java/Swing. Os sistemas Web são construídos com as tecnologias VB/ASP, Java/JSP e PHP. Para o processo de desenvolvimento de sistemas *não-mainframe*, a PROCERGS definiu uma metodologia própria que está baseada no Processo Unificado: na abordagem de desenvolvimento OO, no PU, na UML como linguagem de modelagem, e em inúmeras ferramentas CASE de apoio ao processo de desenvolvimento.

6.1.2.4 Práticas de reuso

A empresa definiu e investe num processo de administração de componentes denominado “Administração de Componentes Reutilizáveis (ACR)” com o objetivo de viabilizar e facilitar o processo de reutilização de componentes. A Divisão de Tecnologia e Infra-Estrutura é a responsável por executar as atividades previstas neste processo.

Apesar de ter um processo de reuso definido, ele encontra-se num estágio inicial de assimilação das práticas de reuso de forma integrada com o processo de desenvolvimento praticado pelas equipes de projeto.

Até o momento, o repositório do ACR conta com 27 artefatos reusáveis catalogados. A maioria desses artefatos é composta de componentes de negócio e de infra-estrutura dos sistemas (principalmente componentes de controle de acesso e segurança dos sistemas). Na área de IHC, foi disponibilizado um conjunto de arquivos HTML/JavaScript que servem como *templates* do padrão de interface de aplicações Web da empresa, e um conjunto mais bem trabalhado de componentes escritos em Java/Swing para serem utilizados no desenvolvimento de aplicações GUI desktop dentro do padrão de interface definido pela empresa para este tipo de sistema.

6.1.3 Escopo da aplicação da abordagem

Para validar a abordagem em um cenário de desenvolvimento que possa ser executado por apenas uma pessoa, restringimos o escopo do reuso ao universo do desenvolvimento de sites institucionais do Setor de Sites e Versões da Divisão de Projetos da PROCERGS.

A missão da Divisão de Projetos (DPJ) é de desenvolver sistemas e produtos com qualidade, prazo e custos controlados, de acordo com a metodologia de desenvolvimento da empresa. Tem como objetivo otimizar o uso de recursos técnicos da empresa, produzindo sistemas com qualidade, em tempo hábil e com previsibilidade de tempo e custo.

~~O Setor de Sites e Versões (SSV) A missão do Setor de Sites e Versões (SSV) é desenvolver projetos de sites, de design, de sistemas gerenciais e versões de sistemas. O SSV tem como objetivo otimizar o uso de recursos técnicos da empresa produzindo sites, produtos de design, sistemas gerenciais e versões de sistemas com qualidade, em tempo hábil e com previsibilidade de tempo e custo.~~

A maioria dos produtos produzidos pelo SSV são sites institucionais de órgãos públicos, sites de eventos (tais como o da Expointer) e de campanhas de governo (ex: Campanha do Agasalho), entre outros. Ao todo, desde a criação do setor em 2003, foram desenvolvidas mais de uma centena destes sistemas.

No momento, as equipes de desenvolvimento do SSV totalizam 29 colaboradores entre analistas de sistemas, designers e programadores Web.

6.1.4 Plano de atividades

Neste estudo de caso, as principais atividades desenvolvidas foram:

- Escolher um projeto-piloto
- Participar das etapas de análise deste projeto: análise de requisitos, modelagem de casos de uso, modelagem conceitual.

- Fazer um inventário de padrões de artefatos reusáveis e catalogá-los conforme ~~artefatos reusáveis e catalogá-los conforme~~ proposto pela abordagem (árvores de reificação).
- Fazer análise de reuso dos casos de uso
- Fazer análise de reuso de interações
- Identificar artefatos reusáveis candidatos
- Levantar estimativas de esforço de desenvolvimento nos diversos cenários (sem, com reuso e com reuso de artefatos candidatos)
- Avaliar o impacto no design no que tange a usabilidade e qualidade do software produzido

6.2 ~~Inventário de padrões de artefatos reusáveis~~

~~O inventário foi feito com base nos artefatos comumente utilizados pelas equipes de projeto do Setor de Sites e Versões. Como estes artefatos não estavam documentados no formato proposto pela nossa abordagem, os passos seguintes foram os de documentá-los, catalogá-los no repositório e publicá-los. Devido à limitação de tempo, a documentação se restringiu na identificação dos padrões e na constituição inicial das respectivas árvores de reificação.~~

~~Durante esse inventário também já foi possível identificar um conjunto de *assets* com potencial para serem utilizados no desenvolvimento dos projetos do setor. Esses *assets* foram cadastrados no repositório com a situação de “proposto” (mais informações sobre as situações do ciclo de vida de um *asset* podem ser obtidas na figura 2.4). A tabela 6.1 apresenta um resumo do número de tipos de *assets* publicados e *assets* propostos que foram identificados no inventário e durante o processo de desenvolvimento do sistema. Listagens completas dos *assets* identificados, organizados na forma de padrões de domínio, padrões de caso de uso reificados e padrões concretos de interação, são encontradas nos apêndices B, C e D, respectivamente.~~

Tabela 6.1: ~~Resumo do inventário de *assets* reusáveis.~~

Tipo de asset	Nº assets publicados	Nº assets propostos
Padrão de domínio	2	2
Padrão de caso de uso orientado ao domínio	44	45
Padrão de caso de uso orientado à tarefa	40	2
Padrão de interação	125	0
AUI	44	47
SUI	128	58
CUI	0	157
FUI	187	70
XUI	88	0
Total	562	324

6.3 Descrição do sistema a ser desenvolvido

O projeto escolhido para a aplicação deste estudo de caso foi o projeto de desenvolvimento de um novo *site* institucional para uma Secretaria de Estado do Rio Grande do Sul, com nova identidade visual, novo layout e estrutura, e com ferramentas para administração do conteúdo dinâmico.

O *site* deverá ser dinâmico, de fácil atualização e gerenciamento, de fácil acesso às informações sobre os serviços públicos pela população em geral. Também deve servir como um canal de comunicação através da publicação de notícias.

Sistemas na Web com estas características, tipicamente necessitam de *sistemas de gerenciamento de conteúdo* (*content management systems - CMS*) como parte integrante da solução. Desta forma, uma alternativa de solução para o desenvolvimento do sistema poderia ser a incorporação (reuso) de algum CMS disponível no mercado. Entretanto, devido às características e requisitos deste sistema, optou-se por desenvolver na íntegra todas as funcionalidades previstas.

O resultado da análise de requisitos deste sistema identificou um total de 76 casos de uso, sendo que 21 casos de uso representam as funcionalidades que estarão disponíveis no *site* Web para a população em geral e 55 casos de uso devem compor o módulo de administração que será utilizado por pessoas autorizadas pela Secretaria de Estado para a administração do conteúdo do *site*. Todos os atores, casos de uso, relacionamentos de casos de uso, bem como os pacotes definidos para o sistema estão documentados nos diagramas de casos de uso apresentados no apêndice E.

6.4 Identificação dos artefatos reusáveis

O processo de identificação dos artefatos reusáveis foi feito com base nos artefatos comumente utilizados pelas equipes de projeto do Setor de Sites e Versões. Como estes artefatos não estavam documentados no formato proposto pela nossa abordagem, os passos seguintes foram os de documentá-los, catalogá-los no repositório e publicá-los. Devido à limitação de tempo, a documentação se restringiu na identificação dos padrões e na constituição inicial das respectivas árvores de reificação.

Durante esse inventário também já foi possível identificar um conjunto de *assets* com potencial para serem utilizados no desenvolvimento dos projetos do setor. Esses *assets* foram cadastrados no repositório com a situação de “proposto” (mais informações sobre as situações do ciclo de vida de um *asset* podem ser obtidas na figura 2.5). A tabela 6.1 apresenta um resumo do número de tipos de *assets* publicados e *assets* propostos que foram identificados no inventário e durante o processo de desenvolvimento do sistema. Listagens completas dos *assets* identificados, organizados na forma de padrões de domínio, padrões concretos de caso de uso e padrões concretos de interação, são encontradas nos apêndices B, C e D, respectivamente.

Tabela 6.1: Resumo do inventário de *assets* reusáveis.

Tipo de <i>asset</i>	Nº <i>assets</i> publicados	Nº <i>assets</i> propostos
Padrão de domínio	2	2
Padrão de caso de uso orientado ao domínio	11	15
Padrão de caso de uso orientado à tarefa	10	2
Padrão de interação	125	0

<u>AUI</u>	<u>11</u>	<u>17</u>
<u>SUI</u>	<u>128</u>	<u>58</u>
<u>CUI</u>	<u>0</u>	<u>157</u>
<u>FUI</u>	<u>187</u>	<u>70</u>
<u>XUI</u>	<u>88</u>	<u>0</u>
Total	562	321

6.5 Praticando o reuso

Conforme o processo de reuso proposto pela nossa abordagem, as atividades de reuso aplicadas neste estudo de caso foram feitas em paralelo ao processo de desenvolvimento do sistema. Foram executadas as atividades de análise de reuso de casos de uso e de interação, bem como a definição de uma SMUI para o sistema.

No momento da publicação desta dissertação, o desenvolvimento do sistema encontra-se na fase de realização de casos de uso, ou seja, estão sendo realizadas as atividades de projeto OO, construção e testes. Sendo assim, não realizamos a análise de reuso de implementação e uma avaliação geral da aplicação da abordagem após o sistema ter sido implantado. Entretanto, como as fases de modelagem e de apresentação de casos de uso foram executadas integralmente, isto foi suficiente para que pudéssemos aplicar a nossa abordagem de reuso e avaliar o impacto da sua utilização junto à equipe de projeto. Futuramente, pretende-se publicar informações e análises complementares às divulgadas nesta dissertação.

6.5.1 Análise de reuso de casos de uso

Na fase de modelagem de casos de uso do sistema, à medida que os casos de uso foram sendo identificados e modelados, foi realizada (baseada nos critérios definidos na seção 5.5.1.1) a análise de reuso de casos de uso e a avaliação do nível de reuso de cada caso de uso. A execução desta análise foi feita tanto pelo analista de sistemas quanto pelo projetista de IU do projeto. As principais tarefas do analista de sistemas foram a de identificar e modelar os casos de uso e a de prospectar padrões de casos de uso (CUCPs) úteis ao projeto do sistema. O projetista de IU, com base na narrativa dos casos de uso e nos CUCPs associados, fez uma avaliação das IUs dos CRUCPs) úteis ao projeto do sistema. O projetista de IU, com base na narrativa dos casos de uso e nos RUCPs associados, fez uma avaliação das IUs dos RUCPs para determinar se são adequadas para suportar a tarefa interativa descrita no caso de uso.

Como as tecnologias usadas para a construção deste sistema (HTML/DHTML/JavaScript na parte que roda no navegador (*browser*) e PHP/MySQL na parte que roda no servidor) foram definidas antes do início desta atividade, foi possível fazer a avaliação do nível de reuso até o nível de XUI.

O resultado desta análise de reuso de casos de uso é apresentado na lista ator-objetivo da tabela 6.2.

Tabela 6.2: Lista ator-objetivo do sistema em estudo.

Nro Pacote	Objetivo ao nível de tarefa	Prioridade CUCP	Nível de Reuso
4 artigo	Consulta artigo	Consulta {objeto}	9 - FUI
5 artigo	Envia artigo		0 - Nenhum

Nro Pacote	Objetivo ao nível de tarefa	Prioridade	CUCP	Nível de Reuso
9 artigo	Lista artigos publicados		Lista {objeto}	9 - FUI
8 artigo	Lista artigo		Lista {objeto}	9 - FUI
10 artigo	Pesquisa artigo		Pesquisa {objeto}	9 - FUI
7 artigo	Inclui artigo		Inclui {objeto}	9 - FUI
2 artigo	Altera artigo		Altera {objeto}	9 - FUI
6 artigo	Exclui artigo		Exclui {objeto}	9 - FUI
11 artigo	Publica artigo			0 - Nenhum
3 artigo	Cancela publicação de artigo			0 - Nenhum
12 banner	Lista banner		Lista {objeto}	9 - FUI
13 banner	Inclui banner		Inclui {objeto}	9 - FUI
14 banner	Exclui banner		Exclui {objeto}	9 - FUI
15 banner	Publica banner			0 - Nenhum
16 contato	Altera contato		Altera {objeto}	9 - FUI
17 conteudo	Lista conteúdo		Lista {objeto}	9 - FUI
18 conteudo	Visualiza conteúdo			0 - Nenhum
19 conteudo	Inclui conteúdo		Inclui {objeto}	9 - FUI
20 conteudo	Altera conteúdo		Altera {objeto}	9 - FUI
21 conteudo	Exclui conteúdo		Exclui {objeto}	9 - FUI
22 entidade	Lista entidades vinculadas		Lista {objeto}	9 - FUI
23 entidade	Inclui entidade vinculada		Inclui {objeto}	9 - FUI
24 entidade	Exclui entidade vinculada		Exclui {objeto}	9 - FUI
26 foto	Lista galeria de fotos		Lista {objeto}	9 - FUI
25 foto	Visualiza foto			0 - Nenhum
27 foto	Inclui galeria de fotos		Inclui {objeto}	9 - FUI
28 foto	Altera galeria de fotos		Altera {objeto}	9 - FUI
29 foto	Exclui galeria de fotos		Exclui {objeto}	9 - FUI
30 foto	Insere foto			0 - Nenhum
31 foto	Remove foto			0 - Nenhum
73 geral	Busca informação			0 - Nenhum
72 geral	Envia mensagem			0 - Nenhum
76 geral	Consulta Ajuda		Solicita ajuda de uma tarefa	9 - FUI
74 geral	Consulta mapa do site			0 - Nenhum
75 geral	Consulta organograma			0 - Nenhum
32 legislacao	Lista legislação		Lista {objeto}	9 - FUI
33 legislacao	Baixa legislação			0 - Nenhum
34 legislacao	Inclui legislação		Inclui {objeto}	9 - FUI
35 legislacao	Altera legislação		Altera {objeto}	9 - FUI
36 legislacao	Exclui legislação		Exclui {objeto}	9 - FUI
37 legislacao	Mantem categoria de legislação		Mantém {informacao}	9 - FUI
38 link	Lista link		Lista {objeto}	9 - FUI

Nro Pacote	Objetivo ao nível de tarefa	Prioridade	CUCP	Nível de Reuso
39 link	Inclui link		Inclui {objeto}	9 - FUI
40 link	Altera link		Altera {objeto}	9 - FUI
41 link	Exclui link		Exclui {objeto}	9 - FUI
42 link	Mantém categoria de link		Mantém {informacao}	9 - FUI
43 noticia	Lista notícias publicadas		Lista notícia	9 - FUI
44 noticia	Visualiza notícia		Consulta notícia	9 - FUI
45 noticia	Envia notícia			0 - Nenhum
46 noticia	Imprime notícia		Imprime {objeto}	9 - FUI
47 noticia	Lista notícia		Lista {objeto}	9 - FUI
48 noticia	Pesquisa notícia		Pesquisa {objeto}	9 - FUI
49 noticia	Inclui notícia		Inclui {objeto}	9 - FUI
50 noticia	Altera notícia		Altera {objeto}	9 - FUI
51 noticia	Exclui notícia		Exclui {objeto}	9 - FUI
52 noticia	Publica notícia			0 - Nenhum
53 noticia	Cancela publicação de notícia			0 - Nenhum
1 noticia	Mantém categoria de notícia		Mantém {informacao}	9 - FUI
54 relatorio	Lista relatório de gestão		Lista {objeto}	9 - FUI
55 relatorio	Baixa relatório de gestão			0 - Nenhum
56 relatorio	Inclui relatório de gestão		Inclui {objeto}	9 - FUI
57 relatorio	Exclui relatório de gestão		Exclui {objeto}	9 - FUI
58 servico	Lista serviços		Lista {objeto}	9 - FUI
59 servico	Inclui serviço		Inclui {objeto}	9 - FUI
60 servico	Altera serviço		Altera {objeto}	9 - FUI
61 servico	Exclui serviço		Exclui {objeto}	9 - FUI
62 servico	Mantem categoria de serviços		Mantém {informacao}	9 - FUI
63 usuario	Identifica-se para o sistema		Identificar-se para o sistema (Login)	9 - FUI
64 usuario	Altera senha		Altera senha	9 - FUI
65 usuario	Solicita nova senha		Solicita nova senha	9 - FUI
70 usuario	Inicializa senha do usuário		Inicializa senha do usuário	10 - XUI
66 usuario	Lista usuário		Lista usuário	10 - XUI
67 usuario	Inclui usuário		Inclui usuário	10 - XUI
68 usuario	Altera usuário		Altera usuário	10 - XUI
69 usuario	Exclui usuário		Exclui usuário	10 - XUI
71 usuario	Mantém nível de permissão		Mantém nível de permissão	10 - XUI

Após esta análise de reuso, constatou-se que em torno de 77% dos casos de usos foi possível praticar algum tipo de reuso (reuso entre os níveis 2 e 10). Particularmente, os números que nos interessam são aqueles que contemplam o reuso de IHC. Neste aspecto, a análise de reuso também obteve o índice de 77% de possibilidade de reuso de IHC (reuso entre os níveis 9 e 10). Uma análise mais minuciosa sobre o impacto da

nossa abordagem de reuso de IHC no processo de desenvolvimento no que tange à produtividade e à qualidade do software produzido será discutida na seção 6.5.

6.5.2 Análise de reuso de interação

Conforme definido na nossa abordagem, a análise de reuso de interação foi realizada conjuntamente com a atividade de prototipação da IU onde foram executadas as três atividades propostas, a saber: 1) revisão da indicação dos artefatos reusáveis de IU apontados na fase anterior durante o mapeamento da IU com os passos dos casos de uso; 2) identificação de CIPs aplicáveis ao projeto de IU do caso de uso; e 3) verificação do nível de reificação do CIP que poderá ser reusado.

Na revisão das indicações de reuso de IHC (reuso entre os níveis 5 e 10) identificados na tabela 6.2, verificou-se que 96,5% das indicações puderam ser efetivamente reusadas. De fato, apenas os casos de uso 12 (Lista banner) e 13 (Inclui banner) não puderam reusar na íntegra as soluções de IHC associadas aos CUCPs pelo fato de que banners são imagens e os CUCPs continham soluções para a interação com dados textuais, o que demandou uma nova definição de interação para manipular imagens. Entretanto, mesmo nestes casos de uso, houve um aproveitamento parcial da solução de IHC apontado pelo CRUCPs pelo fato de que banners são imagens e os RUCPs continham soluções para a interação com dados textuais, o que demandou uma nova definição de interação para manipular imagens. Entretanto, mesmo nestes casos de uso, houve um aproveitamento parcial da solução de IHC apontado pelo RUCP associado.

A seguir, executou-se a análise de reuso de interação (conforme descrito na seção 5.5.1.2) com a investigação dos CIPs com soluções de interação que pudessem atender às necessidades do projeto de IU, e com a determinação do nível de reuso de cada CIP baseado na tecnologia e na arquitetura do sistema definidos para o SsD. O resultado desta atividade identificou inúmeros CIPs com artefatos reusados nos mais variados níveis de reificação no projeto de IU. Casos de uso até então sem nenhuma indicação de reuso de IHC na análise anterior, passaram a ter pelo menos um artefato de IHC para ser reusado.

Para exemplificar como foi realizado o processo de análise de reuso de interação no projeto do sistema, apresentamos a seguir uma descrição passo a passo deste processo para alguns casos de uso do pacote de notícias (casos de uso 52 e 49 da tabela 6.2).

6.5.2.10 caso de uso 52 - Publica notícia.

Conforme identificado na tabela 6.1, o nível de reuso deste caso de uso é zero, ou seja, nenhum artefato reusável de IHC foi identificado após a análise de reuso de casos de uso.

A narrativa deste caso de uso é apresentada na tabela 6.3.

Tabela 6.3: Narrativa do caso de uso 52 - Publica notícia.

Publica notícia

Ator principal: assessor de comunicação

Fluxo principal

1. O assessor de comunicação seleciona uma ou várias notícias e informa ao sistema

que deseja publicá-las

2. O sistema apresenta as notícias selecionadas e aguarda que o assessor de comunicação complete as informações para publicação
3. O assessor de comunicação fornece as informações para publicação
4. Eventualmente, o assessor de comunicação remove uma notícia do conjunto de notícias a serem publicadas
5. O assessor de comunicação submete as notícias para publicação
6. O sistema publica as notícias conforme definido
7. O caso de uso é encerrado

Fluxo alternativo

- 1a. Nenhuma notícia foi selecionada
 - 1a1. Sistema solicita que seja selecionada pelo menos uma notícia e retorna ao passo 1
 - 1b. Uma ou mais notícias selecionadas já foram publicadas
 - 1a1. Sistema solicita que sejam selecionadas apenas as notícias não publicadas e retorna ao passo 1
 - 4a. Existe somente uma notícia para ser removida
 - 4a1. Sistema solicita confirmação da remoção
 - 4a2. O assessor de comunicação confirma a remoção
 - 4a3. O caso de uso é encerrado
 - 4a2a. O assessor de comunicação não confirma a remoção
 - 4a2a1. O caso de uso retorna ao passo 3
 - 5a. Os dados de publicação informados não são válidos ou estão incompletos
 - 5a1. Sistema comunica este fato ao assessor de comunicação e retorna ao passo 3
-

Com base nas tarefas interativas descritas em cada passo do caso de uso foi elaborado um protótipo da IU e feito uma pesquisa no repositório de CIPs em busca de padrões de interação que pudessem endereçar alguma solução de interação para este caso de uso. O protótipo resultante deste trabalho é apresentado na figura 6.1.

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX E-MAIL SENHA AJUDA SAÍDA

Módulo Administração ▼

Publicar notícias

Notícia a ser publicada	Data e Hora publicação	Destaque	
Título da notícia 1	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 2	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 3	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 4	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 5	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 6	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 7	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 8	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 9	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 10	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 11	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>
Título da notícia 12	20/12/2006 Calend [08:00 ▼]	Nenhum ▼	<input type="button" value="Remover"/>

Notícias

Objeto 2.

Objeto 2.

Objeto 2.

Objeto 2.

Nome do operador:
Fulano de Tal e tal

Figura 6.1: Protótipo de IU do caso de uso “Publica notícia”.

Conforme prescrito na fase de apresentação de casos de uso, cada passo do caso de uso foi mapeado com a respectiva IU que suporta o passo. O apêndice G apresenta estes mapeamentos que caracterizam a especificação de um caso de uso apresentado.

Também foram mapeados os CIPs utilizados em cada espaço de interação que compõem a IU do caso de uso. Por último, foi feita a definição do nível de reificação dos CIPs em que é possível de obter artefatos reusáveis. A tabela 6.4 apresenta uma lista dos CIPs utilizados no projeto de IU do caso de uso 52 - Publica notícia com o resultado da indicação do nível de reificação do CIP.

Tabela 6.4: CIPs utilizados na IU do caso de uso 52 - Publica notícia.

CIP	Nível de Reuso
Action Button	5 – SUI
Alternating Row Colors	9 – FUI
Aplicação Web	9 – FUI
Calendário Web	9 – FUI
Edição de data Web	9 – FUI
Edição de hora Web	9 – FUI
Editar Web	9 – FUI
K4 Action Buttons	5 – SUI
Lista sem paginação	9 – FUI
Lista zebraada	9 – FUI

CIP	Nível de Reuso
Mensagem de erro Web	9 – FUI
Mensagem informativa Web	9 – FUI
Pergunta com respostas simples Web	9 – FUI

6.5.2.2O caso de uso 49 – Inclui notícia.

Conforme identificado na tabela 6.1, o nível de reuso deste caso de uso é 9, ou seja, podem-se utilizar artefatos reusáveis em nível de FUI.

A narrativa do caso de uso - baseada na narrativa do **CRUCP** “Inclui {objeto}” - é apresentada na tabela 6.5.

Tabela 6.5: Narrativa do caso de uso 49 – Inclui notícia.

Inclui notícia

Ator principal: assessor de comunicação

Fluxo principal

1. O assessor de comunicação solicita cadastrar uma notícia
2. O sistema apresenta um formulário para preenchimento dos dados da notícia
3. O assessor de comunicação preenche os dados
4. Eventualmente, o assessor de comunicação visualiza a notícia sendo cadastrada
5. O assessor de comunicação conclui o preenchimento e solicita ao sistema que armazene os dados
6. O sistema valida os dados do usuário
7. O sistema armazena os dados informados
8. O caso de uso é encerrado

Fluxo alternativo

3a. O assessor de comunicação preenche alguns dados, mas desiste de cadastrá-los

3a1. Sistema informa que os dados fornecidos serão perdidos e solicita a confirmação da desistência de cadastramento

3a2. O assessor de comunicação confirma a desistência

3a3. O caso de uso é encerrado

3a2a. O assessor de comunicação não confirma a desistência

3a2a1. O caso de uso retorna ao passo 3

6a. Os dados informados não são válidos ou estão incompletos

6a1. Sistema comunica que os dados são inválidos ou incompletos e retorna ao passo 3

Com base nas tarefas interativas descritas em cada passo do caso de uso, foi feito um esboço da IU e uma pesquisa no repositório de CIPs em busca de possíveis soluções de padrões de interação que atendam. Este trabalho resultou no protótipo apresentado nas figuras 6.2, 6.3 e 6.4.

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX E-MAIL SENHA AJUDA SAÍDA

Módulo Administração ▼

Nova Notícia

Salvar Fechar Visualizar Ajuda

Dados da notícia | Categorias e anexos | Notícias relacionadas

Preencha os campos abaixo para criar a notícia (campos em negrito são obrigatórios):

Título da notícia:

Texto da notícia:

Link N I A

Sinopse da notícia:

Autor:

Fonte da Notícia:

Nome do operador:
Fulano de Tal e tal

Figura 6.2: Protótipo de IU do caso de uso “Inclui notícia” - parte 1.

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX E-MAIL SENHA AJUDA SAÍDA

Módulo Administração ▼ **Nova Notícia**

Salvar Fechar Visualizar Ajuda

Dados da notícia | **Categorias e anexos** | Notícias relacionadas

Notícias:

Objeto 2.
Objeto 2.
Objeto 2.
Objeto 2.

Nome do operador:
Fulano de Tal e tal

Categorias:

Adicionar... Remover

Anexo(s):

	Nome do arquivo	Tipo
<input type="checkbox"/>	XXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX
<input type="checkbox"/>	XXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX
<input type="checkbox"/>	XXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX
<input type="checkbox"/>	XXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX
<input type="checkbox"/>	XXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXX

Adicionar... Remover

↑
↓

Figura 6.3: Protótipo de IU do caso de uso “Inclui notícia” - parte 2.

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX E-MAIL SENHA AJUDA SAÍDA

Módulo Administração ▼ **Nova Notícia**

Salvar Fechar Visualizar Ajuda

Dados da notícia | Categorias e anexos | **Notícias relacionadas**

Notícias:

Objeto 2.
Objeto 2.
Objeto 2.
Objeto 2.

Nome do operador:
Fulano de Tal e tal

Notícias relacionadas:

Adicionar... Remover

Figura 6.4: Protótipo de IU do caso de uso “Inclui notícia” - parte 3.

Conforme identificado na fase de apresentação de casos de uso, cada passo do caso de uso foi mapeado com a respectiva IU que suporta o passo. Também foram mapeados os CIPs utilizados em cada espaço de interação que compõe a IU do caso de uso.

Por último, foi feita a definição do nível de reificação dos CIPs em que seria possível reusar. A tabela 6.6 apresenta uma lista dos CIPs utilizados no projeto de IU do caso de uso 49 – Inclui notícia com o resultado da indicação do nível de reificação do CIP.

Tabela 6.6: CIPs utilizados na IU do caso de uso 49 – Inclui notícia.

CIP	Nível de Reuso
Mensagem informativa Web	9 – FUI
Lista zebra	9 – FUI
Mensagem de erro Web	9 – FUI
Pergunta com respostas simples Web	9 – FUI
Edição de texto multi-linha com formatação Web	9 – FUI
Ordenação manual de lista Web	9 – FUI
Aplicação Web	9 – FUI
Inclusão única	9 – FUI
Associação simples com lista extensa Web	9 – FUI
Associação dupla Web	9 – FUI
K3 Tab Rows	5 – SUI
K4 Action Buttons	5 – SUI
K12 Preventing Errors	5 – SUI
K13 Meaningful Error Messages	5 – SUI
Input Error Message	5 – SUI
Form	5 – SUI
Parts Selector	5 – SUI
Constraint Input	5 – SUI
Tabs	5 – SUI
Action Button	5 – SUI
Liquid Layout	5 – SUI
Alternating Row Colors	5 – SUI

6.6 Considerações sobre os resultados obtidos

Nesta seção, são apresentadas as principais considerações com respeito à aplicação da nossa abordagem no estudo de caso da PROCERGS. São analisadas as limitações da aplicação da abordagem no estudo de caso, a integração da abordagem de reuso com as demais atividades do processo de desenvolvimento, a produtividade da equipe e a qualidade das IUs geradas.

6.6.1 Quanto às limitações do estudo de caso

Por ser um estudo de caso (sendo, portanto de caráter experimental) não foi utilizada nenhuma ferramenta CASE que desse suporte ao processo de reuso. O repositório de artefatos reusáveis foi catalogado em um banco de dados pelo responsável pelo estudo de caso, o qual fazia o papel de “interface do repositório” sempre que a equipe de projeto necessitava usá-lo. Certamente que a disponibilização de uma ferramenta de busca e de catalogação de *assets RAS* automatizaria este trabalho.

Neste estudo de caso as árvores de reificação de IU foram basicamente constituídas por AUIs, SUIs, FUIs e XUIs. As CUIs não puderam ser consideradas por não haver uma boa ferramenta gráfica para a modelagem de IU disponível que gerasse especificações de IU em uma UIDL. Desta forma, não foi possível utilizar e validar a utilidade das CUIs como artefato de reuso. Entretanto, acreditamos que, com o uso de CUIs, as possibilidades de reuso aumentariam devido à característica de independência de tecnologia de implementação que elas proporcionam.

6.6.2 Quanto à integração dos processos de reuso no processo de desenvolvimento

Como a nossa abordagem propõe uma integração das atividades de reuso com um processo de desenvolvimento dirigido por casos de uso, as atividades de análise de reuso executadas neste estudo de caso mostraram-se adequadas e integradas com as demais atividades do projeto. O impacto da adição destas atividades no processo foi mínimo, pois o processo de trabalho que a equipe de projeto está acostumada a executar não foi alterado significativamente.

Em particular, a execução das atividades de reuso por profissionais de diferentes áreas de conhecimento mostrou-se muito útil na medida em que as decisões de reuso foram tomadas por pessoas com perfil profissional, aderência à metodologia e conhecimento mais adequado para cada fase do projeto. Na análise de reuso de casos de uso, por exemplo, foi o analista de sistemas quem identificou os padrões de domínio e de casos de uso, mas a avaliação do nível de reuso nos níveis de interação e de implementação na árvore de reificação foi feita, respectivamente, pelo projetista de IU e pelo desenvolvedor. O mesmo ocorreu com a análise de reuso de interações: o projetista de IU atuou na identificação e reuso de CIPs somente até o nível de interação, ficando a cargo do projetista OO e do desenvolvedor avaliarem as possibilidades de reuso da implementação destes CIPs.

A proposta de hierarquizar artefatos reusáveis em níveis de reificação também se mostrou propícia para a identificação de novos artefatos de reuso. Já no trabalho de inventário de artefatos reusáveis foi possível identificar novos artefatos à luz do modelo da árvore de reificação de IU. Também durante a execução das análises de reuso, houve várias situações em que as pessoas faziam comentários do tipo “seria bom se tivéssemos este artefato aqui” o que demonstra que as pessoas assimilaram o modelo mental da árvore de reificação de IU e conseguiram identificar novas “ramificações” (extensões na árvore) ao confrontarem as suas necessidades com o que havia disponível no repositório.

6.6.3 Quanto à produtividade no processo de desenvolvimento

A produtividade do desenvolvimento do sistema deste estudo de caso foi medida em termos de avaliação dos ganhos obtidos na redução do esforço de desenvolvimento pela

prática de reuso da nossa abordagem. Para isso, elaboramos uma proposta de estimativa de esforço de desenvolvimento para projetos com reuso cuja fundamentação conceitual será apresentada na seção 6.5.3.1. A seção 6.5.3.2 apresenta o resultado desta estimativa aplicado ao sistema do estudo de caso, bem como faz algumas considerações com respeito ao impacto que a prática de reuso da nossa abordagem gera na produtividade da equipe de desenvolvimento.

6.6.3.1 Estimativa de esforço de desenvolvimento com reuso: fundamentação conceitual

Para avaliar o impacto da aplicação da nossa abordagem de reuso quanto à redução do esforço de desenvolvimento do sistema (e o conseqüente aumento na produtividade do processo), tomamos como base o método de estimativa de pontos de caso de uso (*UCP – Use Case Points*) de Gustav Karner [Karner 1993].

Através da UCP, é possível estimar projetos em suas fases iniciais, uma vez que o levantamento de casos de uso é feito no início do desenvolvimento do projeto, segundo a metodologia de processo unificado.

Embora esta seja uma técnica de estimativa recente, tem sido utilizada em larga escala pelas empresas que adotam o processo de desenvolvimento unificado e a UML. Também no meio acadêmico é possível encontrar diversos trabalhos e artigos sobre pontos de casos de uso.

Segundo Karner, os pontos de um caso de uso são obtidos a partir da fórmula constante na Figura 6.5:

$$UCP = (UUCW + UAW) \times TCF \times EF$$

onde:

- UCP - Use Case Points*: pontos de casos de uso.
- UUCW - Unadjusted Use Case Weight*: peso não ajustado dos casos de uso.
- UAW - Unadjusted Actor Weight*: peso não ajustado de atores
- TCF - Technical Complexity Factor*: complexidade de fatores técnicos
- EF - Environment Factor*: complexidade de fatores ambientais

Figura 6.5: Fórmula para obter os pontos por caso de uso

A fórmula proposta pelo método UCP não explicita uma variável que represente o ganho (ou perda) obtido com a prática de reuso de software no desenvolvimento do caso de uso. Na verdade, o reuso de código somente é considerado na fórmula de Karner como um dos 13 fatores técnicos que compõem o TCF.

Considerando o impacto que a prática de reuso de software gera na produtividade da equipe de desenvolvimento (com a conseqüente redução de esforço de desenvolvimento), a simples utilização deste fator não é suficiente para estimar o esforço de desenvolvimento em projetos que praticam o reuso.

De forma a incorporar a prática de reuso na estimativa de esforço de desenvolvimento baseado em pontos de casos de uso, propomos estender a fórmula de

Karner através da multiplicação do UCP por um fator de reuso, conforme ilustrado na Figura 6.6.

$$EUCP = UCP \times (1 - RF)$$

onde:

EUCP – *Extended Use Case Point*: pontos de caso de uso estendido

RF – *Reuse Factor*: fator de reuso.

Figura 6.6: Fórmula para obter os pontos por caso de uso em projetos com reuso

O fator de reuso (*Reuse Factor* - *RF*) indica a proporção do esforço de desenvolvimento do caso de uso que foi economizado pela prática de reuso. Assim, se $RF=0$, então em todo o processo de desenvolvimento do caso de uso não é utilizado nenhum artefato de reuso. Qualquer valor de $RF > 0$ indica que algum artefato será reusado e isto, em tese, deve implicar em alguma redução do esforço de desenvolvimento deste caso de uso. Se $RF=1$, então o caso de uso foi totalmente obtido com artefatos de reuso, tal como ocorre na situação em que se identifica nível de reuso 10 (*XUI*) para um caso de uso na análise de reuso de casos de uso.

Nos baseamos na identificação dos artefatos reusáveis que serão utilizados no processo de desenvolvimento do caso de uso para realizar o cálculo do *RF*. Para isso, é necessário conhecer que artefatos são esses e estimar o percentual de esforço de desenvolvimento que esses artefatos economizariam ao serem reutilizados.

Atribuir um valor confiável à *RF* não é uma tarefa trivial tendo em vista que isto deve ser feito no início do projeto do software e, conseqüentemente, não existe um amplo conjunto de informações sobre o projeto para subsidiar esta tarefa. Também, a grande diversidade de tipos de artefatos utilizados no processo de desenvolvimento de um software e a dificuldade em medir o impacto que o reuso destes artefatos geraria em termos de economia de esforço de desenvolvimento são um desafio a ser superado para poder determinar o *RF*.

Sem ter a pretensão de propor um método definitivo e completo para o cálculo de estimativa de *RF* – e considerando o enfoque de reuso em IHC, a aplicação da nossa abordagem de reuso dirigido por casos de uso, e o escopo deste estudo de caso – propomos uma abordagem de estimativa de *RF* baseada nos dados obtidos na atividade de análise de reuso de casos de uso.

Basicamente, os passos para a obtenção do *RF* e o cálculo do *EUCP* são os seguintes: 1) definir a proporção de esforço de desenvolvimento que representa cada fase do processo de desenvolvimento; 2) estabelecer o nível de reuso do caso de uso; 3) avaliar o impacto do reuso em cada fase do processo de desenvolvimento; e 4) obter o *RF* e o *EUCP*. Estes passos serão detalhados a seguir:

- ~~Definir a proporção de esforço de desenvolvimento que representa cada fase do processo de desenvolvimento~~
- ~~Estabelecer o nível de reuso do caso de uso.~~
- ~~Avaliar o impacto do reuso em cada fase do processo de desenvolvimento~~

- ~~Obter o RF e o EUCP~~

~~Estes passos serão detalhados a seguir.~~

- *Definir a proporção de esforço em cada fase do processo de desenvolvimento*

Conhecer os artefatos que serão reusados no processo de desenvolvimento de um caso de uso não é suficiente para determinar o RF. É preciso saber o quanto de esforço de desenvolvimento estes artefatos de fato vão economizar na execução da tarefa que produz o artefato a qual o artefato reusável está suprimindo e nas tarefas que a precedem.

Considerando que cada uma das fases do processo de desenvolvimento tem como saída um determinado tipo de artefato (por ex: a saída da análise é um caso de uso com narrativa completa, um modelo de domínio e um modelo de análise) e que a utilização de artefatos reusáveis pode levar à substituição total ou parcial deste artefato, é possível avaliar o quanto este artefato reusável representaria em termos de economia de esforço nesta fase caso ele não fosse reusado.

A divisão do esforço por fases é feita atribuindo um percentual de esforço médio de cada fase em relação ao esforço total. Este percentual pode ser obtido através de levantamento dos tempos de projetos já executados (obviamente considerando sempre projetos de mesmo tipo e porte).

Para avaliar o impacto de reuso em uma determinada fase do processo, é preciso dividir os pontos de caso de uso por fase conforme demonstrado na fórmula da Figura 6.7.

$$UCP_n = UCP \times EP_n$$

onde:

n : n-ésima fase do processo de desenvolvimento

EP_n *Effort Proportion*: proporção de esforço de desenvolvimento da n-ésima fase do processo de desenvolvimento.

Figura 6.7: Fórmula de cálculo de pontos por caso de uso por fase do processo de desenvolvimento

Assim, um UCP seria o resultado do somatório dos UCPs de cada fase do processo de desenvolvimento.

Toda a organização que desenvolve software de forma sistemática e repetitiva consegue métricas que apontam a proporção de esforço em cada fase do processo de desenvolvimento em relação ao esforço total de desenvolvimento de um determinado tipo de software. Na PROCERGS, historicamente, o esforço para o desenvolvimento de sistemas transacionais de bancos de dados segue a proporção apresentada na Tabela 6.7.

Tabela 6.7: Proporção de esforço de desenvolvimento na PROCERGS.

Fase de desenvolvimento	EP
Análise	0,2
Projeto	0,15
Construção	0,4

Fase de desenvolvimento	EP
Testes	0,1
Gerência	0,15
Total	1

Assim, na PROCERGS, um caso de uso com $UCP = 70$, por exemplo, teria um $UCP_{Análise} = 70 \times 0,2 = 14$, um $UCP_{Projeto} = 70 \times 0,15 = 10,5$ e assim por diante.

- *Estabelecer o nível de reuso do caso de uso.*

Na análise de reuso de casos de uso, ao identificar um CUCP candidato a reuso, faz-se uma avaliação para determinar o nível de reuso através da inspeção da árvore de reificação do CRUCP candidato a reuso, faz-se uma avaliação para determinar o nível de reuso através da inspeção da árvore de reificação do RUCP (ver seção 5.5.1.1). Com o nível de reuso definido, identifica-se os artefatos que serão reusados ao longo do processo de desenvolvimento do caso de uso.

Assim, para um caso de uso que obtém nível de reuso 3 (Narrativa Completa) sabe-se que o principal artefato reusável será o que contém a narrativa do caso de uso e nenhum artefato reusável será utilizado nas etapas subseqüentes do processo de desenvolvimento deste caso de uso (esta afirmativa é válida no que foi investigado até o momento, mas poderá mudar após as análises de reuso futuras). Neste caso, a economia de esforço de desenvolvimento se concentra na fase de modelagem deste caso de uso. Em outra situação, se um caso de uso obtém nível de reuso 5 (SUI), então se sabe que o principal artefato reusável será um protótipo da IU. Logo, haverá economia de esforço de desenvolvimento não só na fase de modelagem, mas também na fase de apresentação do caso de uso. Como último exemplo, se um caso de uso obtém nível de reuso 9 (FUI), então se sabe que o artefato reusável será um código-fonte e, conseqüentemente, a economia de esforço de desenvolvimento poderá se estender até a fase de construção do caso de uso.

- *Avaliar o fator de reuso em cada fase do processo de desenvolvimento*

Dependendo da qualidade e da adequação do artefato reusável ao caso de uso sendo analisado, ele pode produzir pouco ou muito impacto na redução do esforço de desenvolvimento deste caso de uso. Por exemplo, para um caso de uso com nível de reuso 9 (FUI) é possível que ocorram as seguintes situações: o código-fonte do CRUCP será reusado quase sem nenhuma alteração pela equipe de desenvolvimento (situação A); ou os desenvolvedores consideraram que o código-fonte é uma boa base para construir o caso de uso, mas que ainda haverá um volume considerável de código a ser escrito até que o caso de uso esteja construído (situação B). Na situação A, o reuso produzirá um alto impacto na redução do esforço de construção e também nas fases que a antecedem (fases de análise e projeto). Já na situação B, este reuso pode ter baixo impacto na redução do esforço de construção e médio ou alto impacto de redução nas fases anteriores.

A intensidade do impacto do reuso quanto à redução do esforço de desenvolvimento determina um valor para o fator de reuso (*RF*) que definimos conforme apresentado na tabela a seguir.

Tabela 6.8: Tabela de fator de reuso.

Intensidade do impacto	Fator de Reuso (RF)	Comentário
Muito Alta	0,8	Não é preciso fazer nenhum ou quase nenhum esforço adicional no artefato após a adoção do artefato reusável
Alta	0,6	É preciso fazer algum esforço adicional no artefato após a adoção do artefato reusável
Média	0,4	A adoção do artefato reusável reduz no máximo em 50% o esforço de desenvolvimento do artefato.
Baixa	0,2	A adoção do artefato reusável, mesmo sendo aplicável, reduz pouco o esforço de desenvolvimento do artefato.
Muito baixa	0	A adoção do artefato reusável não reduz significativamente o esforço de desenvolvimento do artefato.

Conforme a Tabela 6.8, a mais alta intensidade de impacto de reuso produz um fator de reuso de 0,8. Isto significa que o reuso eliminará, no máximo, 80% de esforço de desenvolvimento. Na prática até poderá reduzir mais que isso, mas não consideramos um fator de reuso de 100% (=1) por entender que por mais que um artefato reusável seja adequado a um caso de uso, sempre haverá algum esforço de configuração e de revisão deste artefato que deverá ser contabilizado no esforço de desenvolvimento do caso de uso.

Para cada caso de reuso, o RF deve ser estimado para cada fase do processo de desenvolvimento. O RF final do caso de reuso será aquele obtido pelo somatório dos RFs ponderados pelo EP de cada fase conforme demonstra a fórmula na Figura 6.8.

$$RF = \sum (RF_n \times EP_n)$$

onde:

n : n-ésima fase do processo de desenvolvimento.

Figura 6.8: Fórmula de cálculo de Fator de Reuso (RF).

Assim, considerando que o caso de uso com nível de reuso 9 do exemplo acima tenha um UCP = 70; que será utilizado o EP_n da PROCERGS; e que foram avaliados e atribuídos os RFs de cada fase de desenvolvimento para as situações A e B; pode-se agora calcular o RF e EUCP deste caso de uso. Os dados e resultados destes cálculos são demonstrados na Tabela 6.9.

Tabela 6.9: Exemplo de cálculo de RF e EUCP.

Fase de desenvolvimento	EP	UCP	Situação A		Situação B	
			RF A	EUCP A	RF B	EUCP B
Análise	0,20	14,0	0,8	2,8	0,8	2,8
Projeto	0,15	10,5	0,8	2,1	0,2	8,4
Construção	0,40	28,0	0,8	5,6	0,2	22,4
Testes	0,10	7,0	0,0	7,0	0,0	7,0
Gerência	0,15	10,5	0,0	10,5	0,0	10,5
Total	1,00	70,0	0,6	28,0	0,27	51,1

Fase de desenvolvimento	EP	UCP	Situação A		Situação B	
			RF A	EUCP A	RF B	EUCP B
Análise	0,20	14,0	0,8	2,8	0,8	2,8

Como os resultados dos cálculos de EUCP do caso de uso de exemplo demonstram na Tabela 6.9, um desenvolvimento **sem** reuso que demandaria 70 pontos de casos de uso, passaria a demandar 28 pontos se fosse praticado o reuso na situação A, e 51,1 pontos de casos de uso se fosse praticado o reuso na situação B. Neste exemplo, os ganhos de desenvolvimento **com** reuso foram na ordem de 60% e 27%, respectivamente, em relação ao desenvolvimento **sem** reuso.

Um aspecto importante no uso do método de estimativa de esforço por pontos de casos de uso estendido é que se o RF não for estimado e a equipe de projeto for efetivamente exercer alguma prática de reuso, então o resultado do cálculo de EUCP sempre estimará um esforço de desenvolvimento maior do que o realmente necessário para este projeto, pois contabilizará como esforço de desenvolvimento aquilo que será obtido mediante o reuso.

Além disso, não considerar RF nas métricas dos projetos (nos levantamentos de tempo feitos “*a posteriori*” para verificar o que efetivamente foi gasto nos projetos) também pode gerar distorções na “calibragem” do tempo médio de desenvolvimento de um ponto de caso de uso, pois se houve prática de reuso nos projetos sendo medidos e essa prática não foi contabilizada, então a métrica pode equivocadamente contabilizar como esforço de desenvolvimento *sem reuso* um tempo bem menor do que aquele realmente necessário caso o projeto tivesse sido realmente desenvolvido sem reuso.

Isto posto, conclui-se que a obtenção de um valor para RF na equação de cálculo de esforço é fundamental para garantir um resultado mais acurado do real esforço necessário para o desenvolvimento de um sistema com reuso e sem reuso.

6.6.3.2 Resultado da estimativa de esforço e considerações quanto à produtividade

Para avaliar os possíveis ganhos de produtividade obtidos com a aplicação da nossa abordagem no desenvolvimento do sistema deste estudo de caso, efetuamos o cálculo de UCP estendido (EUCP) nos cenários de desenvolvimento sem reuso e desenvolvimento com reuso. O cenário de desenvolvimento sem reuso estima o esforço necessário para desenvolver inteiramente o sistema sem nenhuma prática de reuso (RF=0). O cenário de desenvolvimento com reuso calcula o RF com base nos artefatos de IU que efetivamente serão utilizados no projeto.

O resultado deste estudo apontou que para o cenário de desenvolvimento sem reuso são necessários 380 EUCPs para desenvolver o sistema. No cenário de desenvolvimento com reuso, o número de EUCPs cai para 213,2. Portanto, uma redução do esforço de desenvolvimento de aproximadamente 44%.

Os dados utilizados para calcular os EUCPs dos casos de uso deste estudo de caso estão listados no apêndice F.

O leitor mais atento pode ter percebido que o cálculo de esforço de desenvolvimento com reuso apresentado nesta dissertação somente contabiliza o reuso de **CRUCPs** e não o reuso de CIPs. Isto se deve ao fato de que é mais fácil mensurar o impacto de um

artefato reusável em um processo de desenvolvimento dirigido por caso de uso quando ele já está mapeado por caso de uso e, principalmente, porque o representa em algum grau de reificação.

No caso dos CIPs, por eles representarem um conjunto de artefatos reificados de um padrão de interação, somente são identificados na fase de apresentação do caso de uso (durante o projeto de IU) e, portanto, não sendo possível identificá-los antecipadamente na fase de modelagem de casos de uso (lembrando que a estimativa de pontos por casos de uso deve ser feita na fase inicial dos projetos, portanto, sem maiores detalhes do projeto de IU). Além disso, mesmo que fossem identificados seria muito difícil argüir um valor para RF com o seu reuso visto que não se teria um indicador objetivo que indicasse o quanto o reuso do CIP representaria em termos de economia de esforço de desenvolvimento do caso de uso como um todo.

Com isso, o que se pode afirmar é que a redução do esforço de desenvolvimento de um caso de uso será ainda maior do que aquela obtida com o reuso de **CRUCP** se for identificado e utilizado algum CIP após a análise de reuso de interações.

6.6.4 Quanto aos efeitos na qualidade da interação

A prática de reuso de IHC apresentada nesta dissertação não visa somente o aumento da produtividade no processo de desenvolvimento e na melhoria da qualidade do software (no sentido de ausência de falhas na execução do software). O reuso de IHC visa, sobretudo, melhorar a usabilidade do software sendo desenvolvido. Parte-se do pressuposto de que, se boas soluções e práticas de IHC são identificadas e documentadas como padrões, então reusar estas idéias e implementações destas idéias (desde que no mesmo contexto de uso) deve levar o software a ter uma melhor qualidade de IHC.

Apesar do sistema deste estudo de caso, até o presente momento, ainda não ter sido disponibilizado para os usuários - e, portanto, ainda não se pôde fazer uma avaliação ergonômica/teste de usabilidade que pudesse apresentar dados e conclusões mais acuradas sobre a qualidade da interação e, conseqüentemente, sobre o impacto do reuso de IHC neste produto – foi notória a percepção, por parte dos desenvolvedores, de que a utilização dos artefatos reusáveis de IHC durante o projeto, propiciou uma melhora significativa na construção das IUs.

O que pudemos identificar, neste estudo de caso, é que a prática de reuso propiciou melhoras na usabilidade nos seguintes aspectos:

Para a equipe de desenvolvimento, o reuso facilitou enormemente o esforço de implementação das IUs, fazendo com que não fosse gasto tanto tempo e energia na implementação de interações triviais e fosse investido mais tempo no design de interação dos casos de uso mais críticos e/ou complexos. Com isso, esses casos de uso, mesmo sem ter se beneficiado diretamente pelo reuso de IHC, se beneficiaram pela maior dedicação da equipe de desenvolvimento no projeto das IUs que os suportem. Um exemplo disso foi a maior dedicação da equipe no projeto de IU da página principal do site.

O simples envolvimento da equipe de desenvolvimento com padrões de interação, padrões de casos de uso e padrões de domínio por si só já propiciou uma melhora no “*expertise*” da equipe sobre design de interação. Este aporte de conhecimento

inevitavelmente agregou uma maior capacidade da equipe em produzir IUs de melhor qualidade.

A nossa abordagem também propiciou uma maior aplicação de padrões no desenvolvimento do sistema, pois, com a disponibilização de implementações destes padrões, a equipe se sentiu mais inclinada a adotá-los e implementá-los (motivados pelo fato de que as implementações não demandariam tanto esforço de desenvolvimento para incorporar a solução dos padrões no seu projeto).

A qualidade da interação também foi beneficiada pela melhor qualidade do software produzido, pois ao reusar componentes de software largamente testados e depurados, diminuiu, assim, a probabilidade de ocorrência de falhas de execução do sistema.

Do ponto de vista do usuário, a adoção efetiva dos padrões propiciou IUs mais agradáveis e interessantes, mais fáceis de aprender e de lembrar como se usa, pelo simples fato de estarem implementando padrões de soluções de interação já consagradas. Metas de usabilidade como eficácia, eficiência, segurança e utilidade foram mais bem endereçadas por incorporarem as soluções dos padrões de interação no projeto do sistema.

7 CONCLUSÃO

Neste trabalho foi apresentada uma abordagem de reuso que pode contribuir para o reuso de IHC em um processo de desenvolvimento de software baseado na UML. Nossa abordagem prevê o reuso de IHC a partir da especificação de casos de uso, de padrões de casos de uso e de padrões~~um caso de uso e de um padrão~~ de interação.

Para isto, inicialmente, foi feita uma revisão dos conceitos relativos a casos de uso, seus formatos de descrição de narrativa, das características de um processo de desenvolvimento dirigido por casos de uso, e das propostas para a reutilização de casos de uso num processo de desenvolvimento de software baseado na UML.

Considerando os diversos estilos de descrição de cenários de casos de uso abordados se percebe que a modelagem de casos de uso ainda está num nível de maturidade baixo do ponto de vista de padronização do modelo e da sua adequação e integração com os demais artefatos do processo de desenvolvimento de software. Dentro deste quadro, é de nosso entendimento que uma prática de reuso baseado em casos de uso, para ser efetiva, deve passar obrigatoriamente pela adoção de uma padronização das narrativas e dos níveis de detalhamento dos casos de uso e padrões de casos de uso envolvidos.

Também foram revisadas as várias abordagens de reuso que podem contribuir para o reuso de interfaces com o usuário. Atualmente, os esforços para reusar as interfaces com o usuário têm se concentrado mais nas abordagens baseadas em padrões de interação e linguagens de padrões do que em abordagens baseadas em padrões de projeto e componentes. Uma das razões para isso é o fato de que os padrões de interação resolvem um problema de enfoque que as demais abordagens de reuso tem utilizado até então. As abordagens de reuso “tradicionais” (tais como as baseadas em componentes e padrões de projeto) estão claramente focadas em soluções que melhorem a arquitetura do sistema a ser construído. Diferentemente dessas abordagens, os padrões de interação e linguagens de padrões estão focados em reusar soluções que melhorem a usabilidade do sistema em uso [Welie & Trætteberg 2000]. Entretanto, a adoção destas abordagens não tem conduzido a uma mudança significativa na forma com que o software é construído. Na prática, estas técnicas e métodos têm surtido pouco efeito no grau de reusabilidade de um software e, principalmente, na melhoria da produtividade no processo e na qualidade do software produzido.

No estágio atual, os padrões de interação têm sido muito úteis no reuso de conhecimento de projeto de interfaces. Como eles documentam idéias e - na medida que se consolidam - faz-se necessário evoluir para propostas de implementações destas

idéias a serem reusadas. Desta forma, passa-se a praticar, efetivamente, o reuso de software e não só das boas idéias apresentadas nos padrões.

Este trabalho pretende ser uma contribuição para diagnosticar e indicar alternativas para mudar esta situação. Para isto, ele ~~propõe~~^{visa} ~~propon~~ uma abordagem de reuso de artefatos de software interativo dirigida por casos de uso. Possibilidades de reuso são analisadas sempre sob a perspectiva do caso de uso ao longo de todo o seu ciclo de vida.

O foco principal foi o de propor uma solução que estendesse a documentação dos padrões de interação e padrões de casos de uso com a incorporação de árvores de reificação de IU compostas por artefatos que especifiquem e implementem a IU do padrão nos mais variados níveis de abstração/reificação. Com isso, procurou-se criar um procedimento sistemático para implementar alguns destes padrões através do reuso de código e/ou de componentes.

A abordagem de reuso proposta não substitui nem é conflitante com nenhuma outra abordagem de reuso praticada pela indústria de software. De fato, ela é complementar e não excludente com relação às demais. O principal benefício do emprego da nossa abordagem é aquele obtido pela articulação de várias iniciativas de reuso através de um modelo que as consolide em um processo de reuso integrado e ao longo do processo de desenvolvimento de software.

Na prática, a abordagem é uma proposta para articular artefatos reusáveis de variados níveis de abstração de forma que sejam compatíveis e coerentes entre si, propiciando - com esta visão integradora dos artefatos - oportunidades para a identificação e a obtenção de novos artefatos reusáveis através dos processos de reificação, abstração e tradução que incidem sobre cada artefato do repositório.

Apesar da nossa abordagem claramente enfocar o reuso de IHC, isso não significa que os demais componentes do software (objetos de negócio e de infra-estrutura) não foram considerados ou que possa haver algum conflito na aplicação da nossa abordagem com estes componentes “não-IHC”. Pelo contrário, como a abordagem é integrada ao processo de desenvolvimento, ela foi concebida levando em conta a existência e a compatibilidade com os componentes de software como um todo. Além disso, os conceitos e processos aqui propostos também podem ser aplicados para a prática de reuso de outros artefatos além dos artefatos de IHC.

7.1 Contribuições

Esta dissertação é um esforço para a melhoria da prática de reuso de IHC nos projetos de software. As principais contribuições podem ser resumidas nos seguintes pontos:

1. Foi proposta uma extensão do modelo de casos de uso através da explicitação dos fluxos e passos da narrativa do caso de uso (seção 5.3, figura 5.7);
2. Foi proposta a adoção do modelo de casos de uso como representante do modelo de tarefas e um mapeamento dos casos de uso com as IUs que suportam essas tarefas (seção 5.3, figura 5.8);
3. Foi promovida a integração das áreas de Engenharia de Software (ES) e Interação Humano-Computador (IHC) através da explicitação, no processo de desenvolvimento, da fase de apresentação de casos de uso (seção 4.1) e da

participação cooperada de profissionais com perfil mais adequado para a constituição das árvores de reificação de IU (seção 5.1);

4. Foi proposta uma extensão do modelo de desenvolvimento de IUs para aplicações interativas multi-contexto, o Cameleon Framework, através da incorporação de mais dois passos de desenvolvimento no modelo, a saber: *Sketched UI* (SUI) e *Executable UI* (XUI) (seção 5.1, figura 5.1);
5. Foi proposta a utilização do Cameleon Framework para servir de modelo não só de IUs para aplicações interativas multi-contexto, mas como modelo para a organização de artefatos reusáveis de IU através das árvores de reificação de IUs (seção 5.1);
6. Para promover o reuso de IHC - não só de idéias e conceitos - mas também de artefatos reusáveis que possam ser utilizados ao longo de todo o processo de desenvolvimento de um software, foi proposta uma extensão do padrão de interação através do conceito de padrão concreto de interação (CIP) (seção 5.2) e da extensão do padrão de caso de uso através do Padrão Concreto de Caso de Uso (~~C~~padrão de caso de uso reificado (RUCP) (seção 5.3);
7. Foi identificado um tipo especial de IU - a SMUI - e proposto o padrão de SMUI (SMUIP) como IUs que definem a forma de integração, navegação e acesso aos vários casos de uso disponibilizados pelo sistema ao usuário (seção 5.4, figura 5.11);
8. Finalmente, foi proposto um processo de reuso dirigido por casos de uso, integrado e totalmente compatível com processos de desenvolvimento baseados na UML (seção 5.5).

Algumas destas contribuições são originais deste trabalho (por ex: 1, 6 e 8) e outras – para fazer jus à idéia base – foram reusadas de outros trabalhos da literatura (por ex: 2, 3 e 7). No entanto, a integração de todos estes conceitos e técnicas é uma contribuição original desta dissertação.

7.2 Perspectivas de trabalhos futuros

Como perspectivas de continuidade deste trabalho vislumbram-se: a) definição de perfis RAS adequados para a documentação dos CIPs e CUCPs; b) término da implementação da ferramenta de apoio aos processos de análise de reuso de forma integrada às ferramentas usadas no processo de desenvolvimento; c) o aperfeiçoamento, evolução e validação do método de estimativa de esforço de desenvolvimento baseado em pontos de caso de uso (UCP) de forma a considerar o desenvolvimento com reuso no cálculo da estimativa de esforço; d) a extensão da aplicação da abordagem de reuso dirigido por casos de uso e orientada a padrões para os demais artefatos do ciclo de vida do software (artefatos não-IU); e e) integração dRUCPs; ~~b) término da implementação da ferramenta de apoio aos processos de análise de reuso de forma integrada às ferramentas usadas no processo de desenvolvimento; c) o aperfeiçoamento, evolução e validação do método de estimativa de esforço de desenvolvimento baseado em pontos de caso de uso (UCP) de forma a considerar o desenvolvimento com reuso no cálculo da estimativa de esforço; d) a extensão da aplicação da abordagem de reuso dirigido por casos de uso e orientada a padrões para os demais artefatos do ciclo de vida do software (artefatos não-IU); e e) integrar~~ a abordagem de reuso com práticas e ferramentas de gerenciamento de projetos, permitindo alocar e acompanhar tarefas e pessoas em todas

as fases do projeto do software, levando em conta a economia de esforços, tempo e custo obtidos com a prática de reuso.

7.3 Lista de publicações

- | | |
|--------------------------|--|
| [Moreira & Pimenta 2006] | MOREIRA, A.; PIMENTA, M. Reuso de Interfaces Através de Padrões Concretos de Interação. In: Proceedings of Simpósio de Fatores Humanos em Sistemas Computacionais (IHC'06). 2006. |
| [Moreira & Pimenta 2007] | MOREIRA, A.; PIMENTA, M. Casos de (Re)Uso: Uma Abordagem para Reuso de Software Interativo Dirigida por Casos de Uso e Padrões Concretos de Interação. In: Proceedings of VI Jornada Iberoamericana de Ingeniería de Software e Ingeniería del Conocimiento (JIISIC'07). 2007. |

REFERÊNCIAS

- | | |
|-------------------------|--|
| [Abrams et al. 1999] | ABRAMS, M., PHANOURIOU, C., BATONGBACAL, A., WILLIAMS, S., SHUSTER, J. UIML: An Appliance-Independent XML User Interface Language. In <u>Proceedings of the World Wide Web Conference</u> , Toronto. May 1999. Disponível em http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html Acessado em outubro 2006. |
| [Adobe 2007] | Adobe Dreamweaver CS3. Disponível em: < http://www.adobe.com/products/dreamweaver/ > Acesso em: mar. 2007. |
| [Alexander et al. 1977] | ALEXANDER, C.; ISHIKAWA, S.; SILVERSTEIN, M.; JACOBSON, M.; FIKSDAHL-KING, I.; ANGEL, S. A Pattern Language , New York: Oxford University Press, 1977. |
| [Ambler 2006] | AMBLER, S. Types of Reuse In Information Technology. 2006. Disponível em: < http://www.ambysoft.com/essays/typesOfReuse.html >. Acesso em: dez. 2006. |
| [Ambler et al. 2005] | AMBLER, S.; NALBONE, J.; VIZDOS, M. The Enterprise Unified Process: Extending the Rational Unified Process . [S.l.]: Prentice Hall PTR, 2005. |
| [Arhelger et al. 2004] | ARHELGER, A.; HANSON, A.; ERWIN, T. Get started with the AUIML Toolkit. 2004. Disponível em http://www.ibm.com/developerworks/java/library/j-auiml/?Open&ca=daw-ja-news Acessado em abril de 2007. |
| [Barthet 1988] | BARTHET, M. Logiciels Interactifs et Ergonomie - Modèles et méthodes de conception, Dunod Informatique, 1988. |
| [Bezerra 2002] | BEZERRA, E. Princípios de análise e projeto de sistemas com UML . Rio de Janeiro: Elsevier, 2002. |
| [Biddle et al. 2001] | BIDDLE, R.; NOBLE, J.; TEMPERO, E. Essential Use cases and Responsibility in Object-Oriented Development. 2001. Disponível em: < http://www.foruse.com/articles/euc-responsibility.pdf >. Acesso em: jul. 2005. |

- [Biddle et al. 2002] BIDDLE, R.; NOBLE, J.; TEMPERO, E. Supporting Reusable Use cases In: Proceedings of the 7th International Conference on Software Reuse: Methods, Techniques, and Tools. pgs. 210-226. 2002.
- [Bittner & Spence 2003] BITTNER, K.; SPENCE, I. **Use Case Modeling**. Boston: Addison-Wesley, 2003.
- [Booch et al. 2000] BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML, Guia do Usuário**. Rio de Janeiro: Campus, 2000.
- [Borchers 1999] BORCHERS, J. Designing Interactive Music Systems: A Pattern Approach, In: Proceedings of 8th International Conference on Human-Computer Interaction. 1999. pp.276–280.
- [Borchers 2000] BORCHERS, J. A Pattern Approach to Interaction Design In: Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods and Techniques (DIS 2000). Pp. 369-378. 2000.
- [Bradac 1998] BRADAC, M.; FLETCHER, B. A Pattern Language for Developing Form Style Windows, In: Proceedings of Pattern Languages of Program Design 3. pp.347–357. 1998.
- [Breedvelt-Schouten et al. 1997] BREEDVELT-SCHOUTEN, I.; PATERNO, F.; SEVERIJNS, C. Reusable Structures in Task Models. In: Proceedings of Design, Specification and Verification of Interactive Systems (DSVIS'97). pp.225–238. 1997.
- [Brighton 2001] BRIGHTON USABILITY GROUP The Brighton Usability Pattern Collection. 2001. Disponível em: <
<http://www.cmis.brighton.ac.uk/Research/patterns/home.html> >. Acesso em: dez. 2005.
- [Brown 1988] BROWN C. **Human-Computer Interface Design Guidelines**. Norwood: Ablex Publishing, 1988.
- [Calvary et al. 2003] CALVARY, G.; COUTAZ, J.; THEVENIN, D.; LIMBOURG, Q.; BOUIL-LON, L.; VANDERDONCKT, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers, vol. 15, no. 3, 289–308. 2003.
- [CC 2006] CREATIVE COMMONS Commons Legal Code Attribution 2.5. Disponível em
<http://creativecommons.org/licenses/by/2.5/legalcode> Acessado em outubro 2006.
- [Cockburn & Fowler 1998] COCKBURN, A.; FOWLER, M. Question time! about use cases. In: Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '98. v. 33 n. 10. 1998.
- [Cockburn 2005] COCKBURN, A. **Escrevendo Casos de Uso Eficazes**. Porto Alegre: Bookman, 2005.

- | | |
|---|--|
| [Coenraets 2004] | COENRAETS, C. An overview of MXML: The Flex markup language. 2004. Disponível em
 http://www.adobe.com/devnet/flex/articles/paradigm.html
 Acessado em maio de 2007. |
| [Constantine & Lockwood 1999] | CONSTANTINE, L.; LOCKWOOD, L. Software for use: a practical guide to the models and methods of usage-centered design. [S.l.]: Addison-Wesley, 1999. |
| [Constantine & Lockwood 2001] | CONSTANTINE, L.; LOCKWOOD, L. Structure and style in use cases for user interface design. In: HARMELEN, M. Object Modeling and User Interface Design: Designing Interactive Systems. Boston: Addison-Wesley, 2001. p. 245-279. Disponível em http://www.foruse.com/articles/structurestyle2.htm Acessado em abril de 2007. |
| [Cooper & Reimann 2003] | COOPER, A.; REIMANN, R. About Face 2.0: The Essentials of Interaction Design. Indianapolis: Wiley Publishing, 2003. |
| [Coram & Lee 1996] | CORAM, T.; LEE, J. Experiences — A Pattern Language for User Interface Design. In: Proceedings of Pattern Languages of Program Design. 1996. Disponível em: <
 http://www.maplefish.com/todd/papers/experiences/Experiences.html >. Acesso em: dez. 2005. |
| [Cover Pages 2005] | COVER PAGES. XML Markup Languages for User Interface Definition. Disponível em
 http://xml.coverpages.org/userInterfaceXML.html. Acessado em abril de 2007. |
| [Cox 1990] | COX, B. “Planning the Software Industrial Revolution”, em IEEE Software vol.7 n.6, págs. 25-33. Disponível em
 http://virtualschool.edu/cox/pub/PSIR/ Acessado em: abril 2006. |
| [Cox 1992] | COX, B. “What if there is a silver bullet... and the competition gets it first?” em Dr. Dobb’s Journal, Oct 1992. Disponível em
 http://virtualschool.edu/cox/pub/92ByteWhatIfSilverBullet/index.html Acessado em abril 2006. |
| [Cox 1993] | COX, K. User Interface Design, Prentice Hall, Simon & Schuster Asia, Singapore, 1993. |
| [Cybulski & Linden 2000] | CYBULSKI, J.; LINDEN, T. Composing Multimedia Artifacts for Reuse. In: Proceedings of Pattern Languages of Program Design 4. pp.461–488. 2000. |
| [CHCI 2007] | Center for Human-Computer Interaction at Virginia Tech. Disponível em http://www.hci.vt.edu/ Acessado em abril de 2007. |
| [Denning 1994] | DENNING, P. A Discipline of Software Architecture. ACM Interactions, V.1, N.1, Jan 1994, pp 55-65. |

- | | |
|---|---|
| [Duyne et al. 2002] | <u>DUYNE, D.; LANDAY J.; HONG, J. The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience. Boston:Addison-Wesley, 2002.</u> |
| [Eclipse 2007] | <u>Eclipse - an open development platform. Disponível em: < http://www.eclipse.org/ > Acesso em: mar. 2007.</u> |
| [Eisenstein et al. 2000] | <u>EISENSTEIN, J., VANDERDONCKT, J., PUERTA, A. Adapting to Mobile Contexts with User-Interface Modeling. In Proceedings of Workshop on Mobile Computing Systems and Applications 2000. Monterey, CA: IEEE Press, December 7-8, 2000.</u> |
| [Forms 2007] | <u>The Forms Working Group. Disponível em http://www.w3.org/MarkUp/Forms/. Acessado em maio de 2007.</u> |
| [Fowler 1997] | <u>FOWLER, M. Analysis Patterns: Reusable Object Models. [S.l.]:Addison-Wesley, 1997.</u> |
| [Frakes & Isoda 1994] | <u>FRAKES, W., B., ISODA, S. Success Factors of Systematic Software Reuse. IEEE Software, Sep, 1994.</u> |
| [Gaffar et al. 2004] | <u>GAFFAR, A. et al. Modeling Patterns for Task Models, In: Proceedings of TAMODIA 2004, pp 99-104. 2004.</u> |
| [Gaffar et al. 2005] | <u>GAFFAR, A., SEFFAH, A., VAN DER POLL, J. HCI Pattern Semantics in XML: a Pragmatic Approach. In: Proceedings of the 2005 workshop on Human and Social Factors of Software Engineering - HSSE 2005. pp. 1-7. 2005.</u> |
| [Gamma et al. 2000] | <u>GAMMA, E. et al. Padrões de Projeto – Soluções reutilizáveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2000.</u> |
| [GNOME 2004] | <u>GNOME. GNOME Human Interface Guidelines 2.0. 2004. Disponível em: < http://developer.gnome.org/projects/gup/hig/2.0 > Acesso em: mar. 2007.</u> |
| [GPL 1991] | <u>GNU GENERAL PUBLIC LICENSE. Version 2, June 1991. Disponível em http://www.gnu.org/copyleft/gpl.html . Acessado em outubro 2006.</u> |
| [IBM 1992] | <u>IBM. Object-Oriented Interface Design: IBM Common User Access Guidelines. Cary: IBM, 1992.</u> |
| [ICSE 2003] | <u>ICSE Workshop on Bridging the Gaps Between Software Engineering and Human-Computer Interaction. Disponível em http://www.se-hci.org/bridging/ Acessado em abril 2006.</u> |
| [Jacobson 2003] | <u>JACOBSON, I. Use Cases Yesterday, Today and Tomorrow. Rational Edge, 2003. Disponível em http://www.ibm.com/developerworks/rational/library/775.html Acessado em abril de 2007.</u> |
| [Jacobson et al. 1992] | <u>JACOBSON, I. et al. Object-Oriented Software Engineering: A Use case Driven Approach. Reading: Addison-Wesley, 1992.</u> |

[Karner 1993]	<u>KARNER, G. "Resource Estimation for Objectory Projects". Objective Systems SF AB (copyright owned by Rational software). Estados Unidos, 1993.</u>
[KDE 2007]	<u>KDE. KDE User Interface Guidelines. 2007. Disponível em: < http://developer.kde.org/documentation/standards/kde/style/basics/index.html > Acesso em: mar. 2007.</u>
[Kleppe et al. 2004]	<u>KLEPPE, A. WARMER, J. BAST, W. MDA Explained. The Model Driven Architecture: Practice and Promise Boston: Addison-Wesley, 2004.</u>
[Kruchten 2000]	<u>KRUCHTEN, P. The Rational Unified Process: An Introduction. [S.l.]: Addison-Wesley, 2000.</u>
[Larman 2004]	<u>LARMAN, C. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao Processo Unificado. 2.ed. Porto Alegre: Bookman, 2004.</u>
[Larsen & Wilber 2005]	<u>LARSEN, G., WILBER, J. Asset lifecycle management for service-oriented architectures. Disponível em http://www-128.ibm.com/developerworks/rational/library/oct05/wilber/ Acessado em abril 2006.</u>
[Lif 1999]	<u>LIF, M. User-interface modelling—adding usability to use cases. International Journal of Human-Computer Studies, v. 50, n. 3, p. 243-262, Mar 1999.</u>
[Luyten & Coninx 2004]	<u>LUYTEN, K., CONINX, K. Uiml.net: an Open Uiml Renderer for the .Net Framework. In: Proceedings of the Fifth International Conference on Computer-Aided Design of User Interfaces CADUI'2004. Funchal, Isle of Madeira: Springer Netherlands, 2004.</u>
[Mahemoff & Johnston 2001]	<u>MAHEMOFF, M.; JOHNSTON, L Usability Pattern Languages: the "Language" Aspect. In: Proceedings of Human-Computer Interaction (Interact '01). pp. 350-358. Disponível em http://mahemoff.com/paper/ Acesso em: nov. 2005.</u>
[Mahemoff 2001]	<u>MAHEMOFF, M. Weaving High-Level and Low-Level Patterns: An Extended Version of the Planet Pattern Language. Technical Report 2001/21, CSSE Dept., University of Melbourne. 2001.</u>
[Mayhew 1992]	<u>MAYHEW, D. Principles and Guidelines in Software User Interface Design. Englewood Cliffs: Prentice Hall, 1992.</u>
[Mcilroy 1968]	<u>MCILROY, M. Mass Produced software components em Software Engineering: Report on a conference by the NATO Science Committee, págs. 138–150.</u>
[Metzker & Reiterer 2002]	<u>METZKER, E.; REITERER, H. Use and Reuse of HCI Knowledge in the Software Development Lifecycle: Existing Approaches and What Developers Thinks In: Proceedings of Proceedings of the IFIP 17th World Computer Congress. 2002.</u>

- | | |
|---|---|
| <u>[Meyer 1997]</u> | <u>MEYER, B. Object-Oriented Software Construction (2nd Ed.) Upper Saddle River: Prentice-Hall, 1997.</u> |
| <u>[Microsoft 1999]</u> | <u>MICROSOFT. Microsoft Windows User Experience. S.l.: Microsoft Press, 1999.</u> |
| <u>[NetBeans 2007]</u> | <u>NetBeans IDE 5.5. Disponível em: <
http://www.netbeans.org/products/ide/ > Acesso em: mar. 2007.</u> |
| <u>[Nielsen & Loranger 2006]</u> | <u>NIELSEN, J.; LORANGER, H. Prioritizing Web Usability. Thousand Oaks: New Riders Publishing, 2006.</u> |
| <u>[Nielsen 1999]</u> | <u>NIELSEN, J. Designing Web Usability: The Practice of Simplicity. Thousand Oaks: New Riders Publishing, 1999.</u> |
| <u>[OMG 2003]</u> | <u>OBJECT MANAGEMENT GROUP. OMG Unified Modeling Language Specification Version 1.5. 2003. Disponível em: <
http://www.omg.org/docs/formal/03-03-01.pdf >. Acesso em: jul. 2005.</u> |
| <u>[OMG 2005]</u> | <u>OBJECT MANAGEMENT GROUP. OMG Reusable Asset Specification, version 2.2. 2005. Disponível em:
http://www.omg.org/docs/formal/05-11-02.pdf Acesso em: abril 2006.</u> |
| <u>[Overgaard & Palmkvist 2005]</u> | <u>OVERGAARD, G.; PALMKVIST, K. Use cases Patterns and Blueprints. Indianapolis: Addison-Wesley, 2005.</u> |
| <u>[Perzel & Kane 1999]</u> | <u>PERZEL, K.; KANE, D. Usability Patterns for Applications on the World Wide Web, In: Proceedings of Pattern Languages of Program Design 1999. 1999.</u> |
| <u>[Phanouriou 2000]</u> | <u>PHANOURIOU, C. UIML: A Device-Independent User Interface Markup Language. Dissertação submetida à Virginia Polytechnic Institute and State University. Setembro de 2000.</u> |
| <u>[Pimenta 2000]</u> | <u>PIMENTA, M. S. TAREFA: Uma abordagem para Engenharia de Requisitos de Sistemas Interativos. In: Proceedings of In: Anales de Terceras Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes Software (IDEAS 2000). Cancún, México. 2000.</u> |
| <u>[PLML 2006]</u> | <u>Pattern Language Markup Language (PLML). Disponível em http://www.hcipatterns.org. Acessado em julho 2006.</u> |
| <u>[Preece et al. 2005]</u> | <u>PREECE, J.; ROGERS, Y.; SHARP, H. Design de interação: além da interação homem-computador. Porto Alegre: Bookman, 2005</u> |
| <u>[Procaccino et al. 2005]</u> | <u>PROCACCINO, J.D. et al. What do software practitioners really think about project success: an exploratory study. The Journal of Systems and Software 78 (2005) 194–203.</u> |

<u>[PROCERGS 2006]</u>	<u>PROCERGS - Cia de Processamento de Dados do Estado do Rio Grande do Sul. Disponível em: < http://www.procergs.rs.gov.br/index.php >. Acesso em: nov. 2006. 2006.</u>
<u>[Puerta & Eisenstein 2002]</u>	<u>PUERTA, A.; EISENSTEIN, J. XIIML: A Universal Language for User Interfaces. RedWhale Software. 2002.</u>
<u>[Ravichandran & Rothenberger 2003]</u>	<u>RAVICHANDRAN, T.; ROTHENBERGER Software Reuse Strategies and Component Markets In Communications of The ACM, Vol. 46 N° 8, pp. 109 a 114</u>
<u>[Retfig 1994]</u>	<u>RETFIG, M. Prototyping for tiny fingers. Communications of the ACM, 37(4), p. 21-27. 1994.</u>
<u>[Riehle & Zullighoven 1995]</u>	<u>RIEHLE, D.; ZULLIGHOVEN, H. A Pattern Language for Tool Construction and Integration Based on the Tools and Materials Metaphor. In: Proceedings of Pattern Languages of Program Design. pp.9-42. 1995.</u>
<u>[Rijken 1994]</u>	<u>RIJKEN, D. The Timeless Way... the design of meaning. SIGCHI Bulletin. Vol. 6, No. 3. PP. 70-79. 1994</u>
<u>[Rothenberger et al. 2003]</u>	<u>ROTHENBERGER, M. A., DOOLEY, K. J., KULKARNI, U. R., NADA, N. Strategies for Software Reuse: A Principal Component Analysis of Reuse Practices , IEEE Transactions on Software Engineering, Vol. 29, No. 09, September, 2003, pp. 825-837.</u>
<u>[Rudd et al. 1996]</u>	<u>RUDD, J., STERN, K., ISENSEE, S. Low vs. High-fidelity Prototyping Debate. ACM Interactions Magazine, January, p. 76-85. 1996.</u>
<u>[Rui & Butler 2003]</u>	<u>RUI, K.; BUTLER, G. Refactoring use case models: the meta-model. In: Proceedings of the twenty-sixth Australasian computer science conference on Conference in research and practice in information technology - Volume 16 CRIPTS '03. 2003.</u>
<u>[Saeki 1999]</u>	<u>SAEKI, M. Reusing Use case Descriptions for Requirements Specification: Towards Use case Patterns In: Proceedings of the Sixth Asia Pacific Software Engineering Conference. 1999.</u>
<u>[Saeki 2000]</u>	<u>SAEKI, M. Patterns and Aspects for Use cases: Reuse Techniques for Use case Descriptions In: Proceedings of the 4th International Conference on Requirements Engineering (ICRE'00). 2000.</u>
<u>[SEESCOA 2002]</u>	<u>SEESCOA Project. Disponível em http://www.cs.kuleuven.ac.be/cwis/research/distrinet/projects/SEESCOA/ Acessado em maio de 2007.</u>
<u>[Seffah et al. 2005]</u>	<u>SEFFAH, A.; GULLIKSEN, J.; DESMARAIS, M. Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle. HCI Series. Vol. 8. 2005. Springer. 391 p.</u>

- | | |
|--|--|
| <u>[Sindre et al. 1995]</u> | <u>SINDRE G.1; CONRADI R.; KARLSSON E.A The REBOOT Approach to Software Reuse Journal of Systems and Software, Volume 30, Number 3, September 1995, pp. 201-212(12)</u> |
| <u>[Souchon & Vanderdonckt 2003]</u> | <u>SOUCHON, N.; VANDERDONCKT, J. A Review of XML-Compliant User Interface Description Languages. Université Catholique de Louvain. 2003.</u> |
| <u>[Stimmel 1999]</u> | <u>STIMMEL, C. Hold Me, Thrill Me, Kiss Me, Kill Me: Patterns for Developing Effective Concept Prototypes. In: Proceedings of Pattern Languages of Program Design 1999. 1999.</u> |
| <u>[Sun 2002]</u> | <u>SUN. Java Look and Feel Design Guidelines. Reading: Addison-Wesley Publishing, 2002. Disponível em: <
http://java.sun.com/products/jlf/ed2/book/index.html > Acesso em: mar. 2007.</u> |
| <u>[Sutcliffe & Carroll 1999]</u> | <u>SUTCLIFFE A.G. & CARROLL J.M. Designing Claims for Reuse in Interactive Systems Design, /International Journal of Human-Computer Studies, /50(3), pp 213-242. 1999.</u> |
| <u>[Tidwell 1998]</u> | <u>TIDWELL, J. Interaction Patterns. In: Proceedings of Pattern Languages of Program Design (PLoP'98). 1998.</u> |
| <u>[Tidwell 2005]</u> | <u>TIDWELL, J. UI Patterns and Techniques. Disponível em: <
http://time-tripper.com/uipatterns/Introduction >. Acesso em: dez. 2005. 2005.</u> |
| <u>[Tracz 1988]</u> | <u>TRACZ, W. Software Reuse Myths, em ACM SIGSOFT Software Engineering Notes vol. 13 no. 1, Janeiro, págs. 17-21. 1988.</u> |
| <u>[Trower 1995]</u> | <u>TROWER, T. The Windows Interface Guidelines for Software Design. Redmond: Microsoft Press, 1995.</u> |
| <u>[UIML 2004]</u> | <u>UIML User Interface Markup Language Specification Working Draft 3.1. March 2004. Disponível em http://www.oasis-open.org/committees/download.php/5937/uiml-core-3.1-draft-01-20040311.pdf. Acessado em outubro 2006.</u> |
| <u>[UsiXML 2006]</u> | <u>USIXML Specification 1.6.4. Disponível em
http://www.usixml.org/index.php?view=page&idpage=5.
Acessado em outubro de 2006.</u> |
| <u>[Vanderdonckt & Bodart 1993]</u> | <u>VANDERDONCKT, J., BODART, F. Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. In: Proceedings of ACM Conference on Human Aspects in Computing Systems InterCHI'93 (Amsterdam, April 24-28, 1993). ACM Press, New York, 424-429. 1993.</u> |
| <u>[Vanderdonckt et al. 2004]</u> | <u>VANDERDONCKT, J., LIMBOURG, Q., MICHOTTE, B., BOUILLON, L., TREVISAN, D., FLORINS, M. UsiXML: a User Interface Description Language for Specifying Multimodal User Interfaces, in Proc. of W3C Workshop on Multimodal Interaction WMI'2004 (Sophia Antipolis, 19-20 July 2004). 2004.</u> |

[Wake 1998]	WAKE, W. Patterns for Interactive Applications. In: Proceedings of Pattern Languages of Program Design (PLoP'98). 1998.
[Welie & Trættemberg 2000]	WELIE, M., TRÆTTEBERG, H. Interaction Patterns In User Interfaces In: Proceedings of the Pattern Languages of Programming (PloP'2000). Disponível em: < http://www.idi.ntnu.no/~hal/publications/design-patterns/PLoP2k-Welie.pdf >. Acesso em: nov. 2005.
[Welie 2001]	WELIE, M. Task-Based User Interface Design . 2001. 205 f. Dissertação de SIKS (the Graduate School for Information and Knowledge Systems).
[Welie 2005]	WELIE, M. ...patterns in Interaction Design. Disponível em: < http://www.welie.com/index.html >. Acesso em: nov. 2006. 2005.
[Welie 2006]	The Parts Selector Interaction Pattern. Disponível em http://www.welie.com/patterns/showPattern.php?patternID=parts-selector . Acessado em outubro 2006.
[Welie et al. 2000]	WELIE, M., VEER, G., ELIËNS, A. Patterns as Tools for User Interface Design. In: Proceedings of International Workshop on Tools for Working with Guidelines. pp. 313-324. 2000.
[Woo & Robinson 2002]	WOO H., ROBINSON, W. Reuse of Scenario Specifications Using an Automated Relational Learner: A Lightweight Approach. In: Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02). 2002.
[XAML 2007]	XAML. Disponível em http://msdn2.microsoft.com/en-us/library/ms747122.aspx Acessado em maio de 2007.
[XICL 2005]	XICL. The XICL Language. Disponível em http://www.lcc.ufrn.br/~lirisnei/xicl/ . Acessado em maio de 2005.
[XIML Forum 2007]	XIML Forum. Disponível em http://www.ximl.org/ Acessado em abril de 2007.
[XUL 2007]	XUL. XML User Interface Language (XUL) Project. Disponível em http://www.mozilla.org/projects/xul . Acessado em abril de 2007.
[Yahoo 2006]	Yahoo Design Pattern Library. Disponível em http://developer.yahoo.com/ypatterns/index.php . Acessado em outubro de 2006.
[Yongbeom & Stohr 1998]	YONGBEOM, K. STOHR, E. Software Reuse: Survey and Research Directions. In: Journal of Management Information Systems, 14, 4, p. 113-147. 1998.
[Zimmermann et al. 2002]	ZIMMERMANN, G.; VANDERHEIDEN, G.; GILMAN, A. Universal Remote Console - Prototyping for the Alternate Interface Access Standard. Trace R&D Center, University of Wisconsin-Madison. 2002.

ANEXO A – O PADRÃO “PARTS SELECTOR”

back to index

Parts Selector

ZOEK EEN NVM KOOPWONING

MINIMALE ZOEKCRITERIA

Provincie: Noord Holland

Streek:

Plaats: AERDENHOUT, AKERSLOOT, ALKMAAR, AMSTELVEEN, **AMSTERDAM**, AMSTERDAM ZUIDOOST

Selectie plaats(en): AMSTERDAM

Prijsklasse (fl.):

Of tussen (fl.) en (fl.):

From www.funda.nl/

Problem Users need to configure an object consisting of several parts.

Use when A complex objects needs to be created by the users or modified later on. The object consists of several parts where the user must select the appropriate combinations of parts. The parts must be independent of each other so that they can be removed or added without consequences for other parts. The number of parts should be more than 10 and can possibly be categorized. The user may not know which parts are available. At some point users may change their minds and remove selected parts. Users need to know which parts are selected and which are available. The number of available parts may change during usage of the application.

Solution **Show all the parts and allow the user to add or remove a part from the selection list.**

Show the list of parts, in categories if applicable, on one side and the list of selected parts on the other side. Place add and remove functionality in between the left and right side. The user can add parts by selecting them and then adding them to the parts list. Consider speed-ups such as double-clicking or dragging to add or remove an item.

Why Showing both the available and selected parts gives the users a complete and instant overview. The visual organization creates a logical task flow from left to right. Because part lists are used the design can easily accommodate new parts or categories.

More Examples This example is taken from Yahoo. Users can personalize their own pages and this screenshot shows how to select the movie-theatres from which they want to see the listings.

Available Theaters

AMC Saratoga 14 Theatre
Almaden Cinema Five
Aquarius
Camera 3
Camera One
Century 20 Great Mall
Century 24
Century Berryessa 10
Century Capitol 16 San Jose
Century Capitol Drive-In Theatre

Your Choices

AMC Mercado 20
Century 21
Century 22
Century 23
Century 25

Add >>

<< Remove

Finished

Concluído

ANEXO B - DTD DE PADRÃO DE INTERAÇÃO

- | | |
|-------------------------|--|
| [Abrams et al. 1999] | ABRAMS, M.; PHANOURIOU, C.; BATONGBACAL, A.; WILLIAMS, S.; SHUSTER, J. UIML: An Appliance-Independent XML User Interface Language, In Proceedings of the World Wide Web Conference, Toronto. May 1999. Disponível em http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html Acessado em outubro 2006. |
| [Alexander et al. 1977] | ALEXANDER, C. et al A Pattern Language , New York: Oxford University Press, 1977. |
| [Ambler 2004] | AMBLER, S. Extending the RUP with the Strategic Reuse Discipline. 2004. Disponível em: < http://www.enterpriseunifiedprocess.com/essays/strategicReuse.html >. Acesso em: dez. 2005. |
| [Barthet 1988] | BARTHET, M. Logiciels Interactifs et Ergonomie -- Modèles et méthodes de conception, Dunod Informatique, 1988. |
| [Bezerra 2002] | BEZERRA, E. Princípios de análise e projeto de sistemas com UML . Rio de Janeiro: Elsevier, 2002. |
| [Biddle et al. 2001] | BIDDLE, R.; NOBLE, J.; TEMPERO, E. Essential Use cases and Responsibility in Object-Oriented Development. 2001. Disponível em: < http://www.foruse.com/articles/euc-responsibility.pdf >. Acesso em: jul. 2005. |
| [Biddle et al. 2002] | BIDDLE, R.; NOBLE, J.; TEMPERO, E. Supporting Reusable Use cases In: Proceedings of the 7th International Conference on Software Reuse: Methods, Techniques, and Tools. pgs. 210-226. 2002. |
| [Bittner & Spence 2003] | BITTNER, K.; SPENCE, I. Use Case Modeling. Boston: Addison-Wesley, 2003. |
| [Booch et al. 2000] | BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML, Guia do Usuário . Rio de Janeiro: Campus, 2000. |
| [Borchers 1999] | BORCHERS, J. Designing Interactive Music Systems: A Pattern Approach, In: Proceedings of 8th International Conference on Human-Computer Interaction. 1999. pp.276-280. |

- [Borchers 2000] BORCHERS, J. A Pattern Approach to Interaction Design In: Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods and Techniques (DIS 2000). Pp. 369-378. 2000.
- [Bradae 1998] BRADAC, M.; FLETCHER, B. A Pattern Language for Developing Form Style Windows, In: Proceedings of Pattern Languages of Program Design 3. pp.347-357. 1998.
- [Breedvelt-Schouten et al. 1997] BREEDVELT-SCHOUTEN, I.; PATERNO, F.; SEVERIJNS, C. Reusable Structures in Task Models. In: Proceedings of Design, Specification and Verification of Interactive Systems (DSVIS'97). pp.225-238. 1997.
- [Brighton 2001] BRIGHTON USABILITY GROUP The Brighton Usability Pattern Collection. 2001. Disponível em: <<http://www.emis.brighton.ac.uk/Research/patterns/home.html>>. Acesso em: dez. 2005.
- [Calvary et. al. 2003] CALVARY, G., COUTAZ, J., THEVENIN, D., LIMBOURG, Q., BOUIL-LON, L., VANDERDONCKT, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers, vol. 15, no. 3, 289-308. 2003.
- [CC 2006] CREATIVE COMMONS Commons Legal Code Attribution 2.5. Disponível em: <http://creativecommons.org/licenses/by/2.5/legalecode> Acessado em outubro 2006.
- [Cockburn 2005] COCKBURN, A. **Escrevendo Casos de Uso Eficazes**. Porto Alegre: Bookman, 2005.
- [Constantine & Lockwood 1999] CONSTANTINE, L.; LOCKWOOD, L. **Software for use: a practical guide to the models and methods of usage-centered design**. [S.l.]: Addison-Wesley, 1999.
- [Coram & Lee 1996] CORAM, T.; LEE, J. Experiences — A Pattern Language for User Interface Design. In: Proceedings of Pattern Languages of Program Design. 1996. Disponível em: <<http://www.maplefish.com/todd/papers/experiences/Experiences.html>>. Acesso em: dez. 2005.
- [Cox 1990] COX, B. “Planning the Software Industrial Revolution”, em IEEE Software vol.7 n.6, págs. 25-33. Disponível em <http://virtualsechool.edu/cox/pub/PSIR/> Acessado em: abril 2006.
- [Cox 1992] COX, B. “What if there is a silver bullet... and the competition gets it first?” em Dr. Dobb’s Journal, Oct 1992. Disponível em <http://virtualsechool.edu/cox/pub/92ByteWhatIfSilverBullet/index.html> Acessado em abril 2006.
- [Cox 1993] COX, K. User Interface Design, Prentice Hall, Simon & Schuster Asia, Singapore, 1993.

[Cybulski & Linden 2000]	CYBULSKI, J.; LINDEN, T. Composing Multimedia Artifacts for Reuse, In: Proceedings of Pattern Languages of Program Design 4. pp.461-488. 2000.
[Denning 1994]	DENNING, P. A Discipline of Software Architecture. ACM Interactions, V.1, N.1, Jan 1994, pp 55-65.
[Duyne et al. 2002]	DUYNE, D.; LANDAY J.; HONG, J. The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience. Boston: Addison-Wesley, 2002.
[Eisenstein et al. 2000]	EISENSTEIN, J.; VANDERDONCKT, J.; PUERTA, A. Adapting to Mobile Contexts with User-Interface Modeling. In: Proceedings of Workshop on Mobile Computing Systems and Applications 2000. Monterey, CA: IEEE Press, December 7-8, 2000.
[Fowler 1997]	FOWLER, M. Analysis Patterns: Reusable Object Models. [S.l.]: Addison-Wesley, 1997.
[Frakes & Isoda 1994]	FRAKES, W., B., ISODA, S. Success Factors of Systematic Software Reuse. IEEE Software, Sep, 1994.
[Gaffar et al. 2004]	GAFFAR, A. et al. Modeling Patterns for Task Models, In: Proceedings of TAMODIA 2004, pp 99-104. 2004.
[Gaffar et al. 2005]	GAFFAR, A., SEFFAH, A., VAN DE POLL, J. HCI Pattern Semantics in XML: a Pragmatic Approach. 2005.
[Gamma et al. 2000]	GAMMA, E. et al. Padrões de Projeto — Soluções reutilizáveis de Software Orientado a Objetos. Bookman, 2000.
[GPL 1991]	GNU GENERAL PUBLIC LICENSE. Version 2, June 1991. Disponível em http://www.gnu.org/copyleft/gpl.html . Acessado em outubro 2006.
[HCSE 2003]	ICSE Workshop on Bridging the Gaps Between Software Engineering and Human-Computer Interaction. Disponível em http://www.se-hci.org/bridging/ Acessado em abril 2006.
[Jacobson et al. 1992]	JACOBSON, I. et al. Object-Oriented Software Engineering: A Use case Driven Approach. Reading: Addison-Wesley, 1992.
[Karner 1993]	KARNER, G. "Resource Estimation for Objectory Projects". Objective Systems SF AB (copyright owned by Rational software). Estados Unidos, 1993.
[Kleppe et al. 2004]	KLEPPE, A. WARMER, J. BAST, W. MDA Explained. The Model Driven Architecture: Practice and Promise Boston: Addison-Wesley, 2004.
[Kruchten 2000]	KRUCHTEN, P. The Rational Unified Process: An Introduction. [S.l.]: Addison-Wesley, 2000.

[Larman 2004]	LARMAN, C. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao Processo Unificado . 2.ed. Porto Alegre: Bookman, 2004.
[Larsen & Wilber 2005]	LARSEN, G., WILBER, J. Asset lifecycle management for service-oriented architectures. Disponível em http://www-128.ibm.com/developerworks/rational/library/oct05/wilber/ Acessado em abril 2006.
[Luyten & Coninx 2004]	LUYTEN, K., CONINX, K. Uiml.net: an Open Uiml Renderer for the .Net Framework. In CADUI'2004. 2004.
[Mahemoff & Johnston 2001]	MAHEMOFF, M.; JOHNSTON, L Usability Pattern Languages: the "Language" Aspect. In: Proceedings of Human-Computer Interaction (Interact '01). pgs 350-358. Disponível em http://mahemoff.com/paper/ Acesso em: nov. 2005.
[Mahemoff 2001]	MAHEMOFF, M. Weaving High-Level and Low-Level Patterns: An Extended Version of the Planet Pattern Language. Technical Report 2001/21, CSSE Dept., University of Melbourne. 2001.
[Meilroy 1968]	MCILROY, M. Mass Produced software components em Software Engineering; Report on a conference by the NATO Science Committee, págs. 138-150.
[Metzker & Reiterer 2002]	METZKER, E.; REITERER, H. Use and Reuse of HCI Knowledge in the Software Development Lifecycle: Existing Approaches and What Developers Thinks In: Proceedings of Proceedings of the IFIP 17th World Computer Congress. 2002.
[Meyer 1997]	MEYER, B. Object-Oriented Software Construction , Prentice-Hall, 2ª Ed. 1997.
[OMG 2003]	OBJECT MANAGEMENT GROUP. OMG Unified Modeling Language Specification Version 1.5 . 2003. Disponível em: < http://www.omg.org/docs/formal/03-03-01.pdf >. Acesso em: jul. 2005.
[OMG 2005]	OBJECT MANAGEMENT GROUP. OMG Reusable Asset Specification, version 2.2 . 2005. Disponível em: http://www.omg.org/docs/formal/05-11-02.pdf Acesso em: abril 2006.
[Overgaard & Palmkvist 2005]	OVERGAARD, G.; PALMKVIST, K. Use cases Patterns and Blueprints . Indianapolis: Addison-Wesley, 2005.
[Perzel & Kane 1999]	PERZEL, K.; KANE, D. Usability Patterns for Applications on the World Wide Web, In: Proceedings of Pattern Languages of Program Design 1999. 1999.
[Phanouriou 2000]	PHANOURIOU, C. UIML: A Device-Independent User Interface Markup Language . Dissertação submetida à Virginia Polytechnic Institute and State University. Setembro de 2000.

[Pimenta 2000]	PIMENTA, M. S. TAREFA: Uma abordagem para Engenharia de Requisitos de Sistemas Interativos. In: Proceedings of Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes Software, Ideas, 2000, Cancún. Memórias... Cuernavaca: Centro Nacional de Investigación y Desarrollo Tecnológico, 2000.
[PLML 2006]	Pattern Language Markup Language (PLML). Disponível em http://www.heipatterns.org . Acessado em julho 2006.
[Preece et al. 2005]	PREECE, J.; ROGERS, Y.; SHARP, H. Design de interação: além da interação homem-computador . Porto Alegre: Bookman, 2005
[Procaccino et al. 2005]	PROCACCINO, J.D. et al. What do software practitioners really think about project success: an exploratory study. The Journal of Systems and Software 78 (2005) 194–203.
[PROCERGS 2006]	PROCERGS – Cia de Processamento de Dados do Estado do Rio Grande do Sul. Disponível em: < http://www.procergs.rs.gov.br/index.php >. Acesso em: nov. 2006. 2006.
[Puerta & Eisenstein 2002]	PUERTA, A.; EISENSTEIN, J. XIML: A Universal Language for User Interfaces. RedWhale Software. 2002.
[Ravichandran & Rothenberger et al. 2003]	RAVICHANDRAN, T.; ROTHENBERGER Software Reuse Strategies and Component Markets In Communications of The ACM, Vol. 46 Nº 8, pp. 109 a 114
[Retfig 1994]	RETFIG, M. Prototyping for tiny fingers. Communications of the ACM, 37(4), p. 21-27. 1994.
[Riehle & Zullighoven 1995]	RIEHLE, D.; ZÜLLIGHOVEN, H. A Pattern Language for Tool Construction and Integration Based on the Tools and Materials Metaphor. In: Proceedings of Pattern Languages of Program Design. pp.9–42. 1995.
[Rijken 1994]	RIJKEN, D. The Timeless Way... the design of meaning. SIGCHI Bulletin. Vol. 6, No. 3. PP. 70–79. 1994
[Rothenberger et al. 2003]	ROTHENBERGER, M. A., DOOLEY, K. J., KULKARNI, U. R., NADA, N. Strategies for Software Reuse: A Principal Component Analysis of Reuse Practices, IEEE Transactions on Software Engineering, Vol. 29, No. 09, September, 2003, pp. 825–837.
[Rudd et al. 1996]	RUDD, J., STERN, K., ISENSEE, S. Low vs. High-fidelity Prototyping Debate. ACM Interactions Magazine, January, p. 76–85. 1996.
[Rui & Butler 2003]	RUI, K.; BUTLER, G. Refactoring use case models: the meta-model. In: Proceedings of the twenty-sixth Australasian computer science conference on Conference in research and practice in information technology – Volume 16 CRIPTS '03. 2003.

[Saeki 1999]	SAEKI, M. Reusing Use-case Descriptions for Requirements Specification: Towards Use-case Patterns In: Proceedings of the Sixth Asia Pacific Software Engineering Conference. 1999.
[Saeki 2000]	SAEKI, M. Patterns and Aspects for Use-cases: Reuse Techniques for Use-case Descriptions In: Proceedings of the 4th International Conference on Requirements Engineering (ICRE'00). 2000.
[Seffah et al. 2005]	SEFFAH, A.; GULLIKSEN, J.; DESMARAIS, M. Human-Centered Software Engineering – Integrating Usability in the Software Development Lifecycle. HCI Series. Vol. 8. 2005. Springer. 391 p.
[Sindre et al. 1995]	SINDRE G.I.; CONRADI R.; KARLSSON E.A The REBOOT Approach to Software Reuse Journal of Systems and Software, Volume 30, Number 3, September 1995, pp. 201-212(12)
[Stimmel 1999]	STIMMEL, C. Hold Me, Thrill Me, Kiss Me, Kill Me: Patterns for Developing Effective Concept Prototypes. In: Proceedings of Pattern Languages of Program Design 1999. 1999.
[Sutcliffe & Carroll 1999]	SUTCLIFFE A.G. & CARROLL J.M. Designing Claims for Reuse in Interactive Systems Design, /International Journal of Human-Computer Studies, /50(3), pp 213-242. 1999.
[Tidwell 1998]	TIDWELL, J. Interaction Patterns In: Proceedings of Pattern Languages of Program Design. 1998.
[Tidwell 2005]	TIDWELL, J. UI Patterns and Techniques. Disponível em: < http://time-tripper.com/uipatterns/Introduction >. Acesso em: dez. 2005. 2005.
[Traez 1988]	TRACZ, W. Software Reuse Myths, em ACM SIGSOFT Software Engineering Notes vol. 13 no. 1, Janeiro, págs. 17-21. 1988.
[Traez 1994]	TRACZ, W. Software Reuse Myths Revisited, In: Proceedings of the 16th International Conference on Software Engineering pgs. 271 – 272. 1994
[UIML 2004]	UIML User Interface Markup Language Specification Working Draft 3.1. March 2004. Disponível em http://www.oasis-open.org/committees/download.php/5937/uiml-core-3.1-draft-01-20040311.pdf . Acessado em outubro 2006.
[UsiXML 2006]	USIXML Specification 1.6.4. Disponível em http://www.usixml.org/index.php?view=page&idpage=5 . Acessado em outubro de 2006.
[Vanderdonckt & Bodart 1993]	VANDERDONCKT, J., BODART, F. Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. In: Proceedings of ACM Conference on Human Aspects in Computing Systems InterCHI'93 (Amsterdam, April 24-28, 1993). ACM Press, New York, 424-429. 1993.

- | | |
|----------------------------|--|
| [Wake-1998] | WAKE, W. Patterns for Interactive Applications . 1998. |
| [Welie & Trættemberg 2000] | WELIE, M., TRÆTTEBERG, H. Interaction Patterns In User Interfaces In: Proceedings of the Pattern Languages of Programming (PloP'2000) . Disponível em: < http://www.idi.ntnu.no/~hal/publications/design-patterns/PLoP2k-Welie.pdf >. Acesso em: nov. 2005. |
| [Welie-2001] | WELIE, M. Task-Based User Interface Design . 2001. 205 f. Dissertação de SIKS (the Graduate School for Information and Knowledge Systems) . |
| [Welie-2005] | WELIE, M. ...patterns in Interaction Design. Disponível em: < http://www.welie.com/index.html >. Acesso em: nov. 2006. |
| [Welie-2006] | The Parts Selector Interaction Pattern. Disponível em: http://www.welie.com/patterns/showPattern.php?patternID=parts-selector . Acessado em outubro 2006. |
| [Welie et al. 2000] | WELIE, M., VEER, G., ELIËNS, A. Patterns as Tools for User Interface Design. In: Proceedings of International Workshop on Tools for Working with Guidelines . pp. 313-324. 2000. |
| [Woo & Robinson-2002] | WOO H., ROBINSON, W. Reuse of Scenario Specifications Using an Automated Relational Learner: A Lightweight Approach. In: Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02) . 2002. |
| [Yahoo-2006] | Yahoo Design Pattern Library. Disponível em: http://developer.yahoo.com/ypatterns/index.php . Acessado em outubro de 2006. |
| [Yongbeom & Stohr 1998] | YONGBEOM, K. STOHR, E. Software Reuse: Survey and Research Directions, In: Journal of Management Information Systems , 14, 4, p. 113-147. 1998. |

anexo – DTD de padrão de interação

Abaixo, segue a especificação em PLML (Pattern Language Markup Language) [PLML 2006] de um padrão de interação.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Dappere Dodo (Dappere Dodo) -->
<!-- Pattern Language Markup Language (PLML) -->
<!-- Change log:

v.1.1.1.2 by Susan 26/05/2003.
- added collection to pattern (for consistency with pattern-link)

v.1.1.1.1 by Jan, Susan 27/04/2003.
- changed order of elements
- added change-log in management

v.1.1 by Martijn 21/04/2003.
- Relaxed datatypes so that pattern-link can be used almost everywhere.
```

- pattern-link was extended to include new attributes collection and label
 - Renamed ID to patternID

v.1.1.0 by Xavier 07/04/2003.
 - Initial draft.

```
-->
<!ELEMENT pattern (name?, confidence?, alias*, synopsis?, illustration?, context?, problem?, forces?, evidence?, solution?, diagram?, implementation?, related-patterns?, pattern-link*, literature?, management?)>
<!ATTLIST pattern
  patternID CDATA #REQUIRED
  collection CDATA #REQUIRED
>
<!ELEMENT name (#PCDATA)>
<!ELEMENT confidence (#PCDATA)>
<!ELEMENT alias (#PCDATA)>
<!ELEMENT synopsis (#PCDATA)>
<!ELEMENT illustration ANY>
<!ELEMENT context EMPTY>
<!ATTLIST context
  mylabel CDATA #IMPLIED>
<!ELEMENT problem (#PCDATA)>
<!ELEMENT forces ANY>
<!ELEMENT evidence (example*, rationale?)>
<!ELEMENT example ANY>
<!ELEMENT rationale ANY>
<!ELEMENT solution ANY>
<!ELEMENT diagram ANY>
<!ELEMENT implementation ANY>
<!ELEMENT related-patterns ANY>
<!ELEMENT pattern-link EMPTY>
<!ATTLIST pattern-link
  type CDATA #REQUIRED
  patternID CDATA #REQUIRED
  collection CDATA #REQUIRED
  label CDATA #REQUIRED
>
<!ELEMENT management (author?, revision-number?, creation-date?, last-modified?, change-log?, credits?)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT creation-date (#PCDATA)>
<!ELEMENT credits (#PCDATA)>
<!ELEMENT revision-number (#PCDATA)>
<!ELEMENT last-modified (#PCDATA)>
<!ELEMENT literature ANY>
```

APÊNDICE A – DTD DE PADRÃO CONCRETO DE INTERAÇÃO

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Concrete Interaction Pattern (CIP) -->
<!-- Change log:

v.1.0 by Augusto Moreira 10/07/2006.
    - Initial draft.
-->
<!DOCTYPE CIP [
<!ELEMENT cip (pattern, ui*)>
<!ENTITY pattern SYSTEM "http://www.hcipatterns.org/dtd/plml.1.00.dtd">
<!ELEMENT ui (aui?, sui, cui*, fui*, xui*)>
<!ELEMENT aui (name, artifact*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT artifact ANY>
<!ELEMENT sui (name, vocabulary, artifact*)>
<!ELEMENT vocabulary (#PCDATA)>
<!ELEMENT cui (name, vocabulary, uidl, artifact*, related-cui*)>
<!ELEMENT uidl (#PCDATA)>
<!ELEMENT related-cui EMPTY>
<!ELEMENT fui (name, language, license?, artifact*)>
<!ELEMENT language (#PCDATA)>
<!ELEMENT license (#PCDATA)>
<!ELEMENT xui (name, configuration?, artifact*)>
<!ELEMENT configuration (operating-system?, hardware?, virtual-machine*)>
<!ELEMENT operating-system (#PCDATA)>
<!ELEMENT hardware (#PCDATA)>
<!ELEMENT virtual-machine (#PCDATA)>

<!ATTLIST cip cipID CDATA #REQUIRED>
<!ATTLIST aui auiID CDATA #REQUIRED>
<!ATTLIST sui suiID CDATA #REQUIRED>
<!ATTLIST cui cuiID CDATA #REQUIRED>
<!ATTLIST fui fuiID CDATA #REQUIRED>
<!ATTLIST xui xuiID CDATA #REQUIRED>
<!ATTLIST cui-link
    type CDATA #REQUIRED
    cuiID CDATA #REQUIRED
>
]>
```

APÊNDICE B – PADRÕES DE DOMÍNIO DO ESTUDO DE CASO

Tabela B.1: Padrões de domínio do estudo de caso.

Padrão de Domínio	Pacotes	Situação
Manutenção de dados cadastrais	crud	Publicado
Publicação de conteúdo na Web	noticias, links	Proposto
Autenticação de usuário e permissões de acesso	accesscontrol	Publicado
Documentação para usuário final (Ajuda)	ajuda	Proposto

APÊNDICE C – PADRÕES CONCRETOS DE CASOS DE USODE CASOS DE USO REIFICADOS DO ESTUDO DE CASO

A tabela a seguir apresenta os assets dos padrões concretos de casos de usode-casos de uso reificados que foram catalogados e utilizados no estudo de caso (assets com a situação: “Publicado”). A tabela também relaciona os assets que foram identificados ao longo do processo de desenvolvimento do sistema como sendo candidatos a serem construídos para futuro reuso (assets com a situação: “Proposto”).

Tabela C.1: Padrões concretos de casos de usode-casos de uso reificados do estudo de caso.

Pacote	Tipo	Asset	Situação
crud	CUCP	Pesquisa {objeto}	Publicado
crud	SUI	Pesquisa {objeto} Web SUI	Publicado
crud	CUI	Pesquisa {objeto} Web CUI Básica	Proposto
crud	FUI	Pesquisa {objeto} Web FUI XHTML/CSS	Publicado
crud	FUI	Pesquisa {objeto} Web FUI VB/ASP	Publicado
crud	FUI	Pesquisa {objeto} Web FUI Java/JSP	Proposto
crud	FUI	Pesquisa {objeto} Web FUI Java/JSP/EJB	Publicado
crud	FUI	Pesquisa {objeto} Web FUI PHP	Publicado
crud	SUI	Pesquisa {objeto} avançado Web SUI	Publicado
crud	CUI	Pesquisa {objeto} avançado Web CUI Básica	Proposto
crud	FUI	Pesquisa {objeto} avançado Web FUI XHTML/CSS	Publicado
crud	FUI	Pesquisa {objeto} avançado Web FUI VB/ASP	Publicado
crud	FUI	Pesquisa {objeto} avançado Web FUI Java/JSP	Proposto
crud	FUI	Pesquisa {objeto} avançado Web FUI Java/JSP/EJB	Publicado
crud	FUI	Pesquisa {objeto} avançado Web FUI PHP	Publicado
crud	SUI	Pesquisa {objeto} Desktop SUI	Publicado
crud	CUI	Pesquisa {objeto} Desktop CUI Básica	Proposto
crud	FUI	Pesquisa {objeto} Desktop FUI Java/Swing	Publicado
crud	FUI	Pesquisa {objeto} Desktop FUI VB	Publicado
crud	SUI	Pesquisa {objeto} avançado Desktop SUI	Publicado
crud	CUI	Pesquisa {objeto} avançado Desktop CUI Básica	Proposto
crud	FUI	Pesquisa {objeto} avançado Desktop FUI Java/Swing	Publicado

Pacote	Tipo	Asset	Situação
crud	FUI	Pesquisa {objeto} avançado Desktop FUI VB	Publicado
crud	CUCP	Lista {objeto}	Publicado
crud	SUI	Lista {objeto} Web SUI	Publicado
crud	CUI	Lista {objeto} Web CUI Básica	Proposto
crud	FUI	Lista {objeto} Web FUI XHTML/CSS	Publicado
crud	FUI	Lista {objeto} Web FUI VB/ASP	Publicado
crud	FUI	Lista {objeto} Web FUI Java/JSP	Proposto
crud	FUI	Lista {objeto} Web FUI Java/JSP/EJB	Publicado
crud	FUI	Lista {objeto} Web FUI PHP	Publicado
crud	SUI	Lista {objeto} Web paginação SUI	Publicado
crud	CUI	Lista {objeto} Web paginação CUI Básica	Proposto
crud	FUI	Lista {objeto} Web paginação FUI XHTML/CSS	Publicado
crud	FUI	Lista {objeto} Web paginação FUI VB/ASP	Publicado
crud	FUI	Lista {objeto} Web paginação FUI Java/JSP	Proposto
crud	FUI	Lista {objeto} Web paginação FUI Java/JSP/EJB	Publicado
crud	FUI	Lista {objeto} Web paginação FUI PHP	Publicado
crud	SUI	Lista {objeto} Desktop SUI	Publicado
crud	CUI	Lista {objeto} Desktop CUI Básica	Proposto
crud	FUI	Lista {objeto} Desktop FUI Java/Swing	Publicado
crud	FUI	Lista {objeto} Desktop FUI VB	Publicado
crud	CUCP	Consulta {objeto}	Publicado
crud	SUI	Consulta {objeto} Web SUI	Publicado
crud	CUI	Consulta {objeto} Web CUI Básica	Proposto
crud	FUI	Consulta {objeto} Web FUI XHTML/CSS	Publicado
crud	FUI	Consulta {objeto} Web FUI VB/ASP	Publicado
crud	FUI	Consulta {objeto} Web FUI Java/JSP	Proposto
crud	FUI	Consulta {objeto} Web FUI Java/JSP/EJB	Publicado
crud	FUI	Consulta {objeto} Web FUI PHP	Publicado
crud	SUI	Consulta {objeto} Desktop SUI	Publicado
crud	CUI	Consulta {objeto} Desktop CUI Básica	Proposto
crud	FUI	Consulta {objeto} Desktop FUI Java/Swing	Publicado
crud	FUI	Consulta {objeto} Desktop FUI VB	Publicado
crud	CUCP	Inclui {objeto}	Publicado
crud	SUI	Inclui {objeto} Web SUI	Publicado
crud	CUI	Inclui {objeto} Web CUI Básica	Proposto
crud	FUI	Inclui {objeto} Web FUI XHTML/CSS	Publicado
crud	FUI	Inclui {objeto} Web FUI VB/ASP	Publicado
crud	FUI	Inclui {objeto} Web FUI Java/JSP	Proposto
crud	FUI	Inclui {objeto} Web FUI Java/JSP/EJB	Publicado
crud	FUI	Inclui {objeto} Web FUI PHP	Publicado

Pacote	Tipo	Asset	Situação
crud	SUI	Inclui {objeto} Desktop SUI	Publicado
crud	CUI	Inclui {objeto} Desktop CUI Básica	Proposto
crud	FUI	Inclui {objeto} Desktop FUI Java/Swing	Publicado
crud	FUI	Inclui {objeto} Desktop FUI VB	Publicado
crud	CUCP	Altera {objeto}	Publicado
crud	SUI	Altera {objeto} Web SUI	Publicado
crud	CUI	Altera {objeto} Web CUI Básica	Proposto
crud	FUI	Altera {objeto} Web FUI XHTML/CSS	Publicado
crud	FUI	Altera {objeto} Web FUI VB/ASP	Publicado
crud	FUI	Altera {objeto} Web FUI Java/JSP	Proposto
crud	FUI	Altera {objeto} Web FUI Java/JSP/EJB	Publicado
crud	FUI	Altera {objeto} Web FUI PHP	Publicado
crud	SUI	Altera {objeto} Desktop SUI	Publicado
crud	CUI	Altera {objeto} Desktop CUI Básica	Proposto
crud	FUI	Altera {objeto} Desktop FUI Java/Swing	Publicado
crud	FUI	Altera {objeto} Desktop FUI VB	Publicado
crud	CUCP	Exclui {objeto}	Publicado
crud	SUI	Exclui {objeto} Web SUI	Publicado
crud	CUI	Exclui {objeto} Web CUI Básica	Proposto
crud	FUI	Exclui {objeto} Web FUI XHTML/CSS	Publicado
crud	FUI	Exclui {objeto} Web FUI VB/ASP	Publicado
crud	FUI	Exclui {objeto} Web FUI Java/JSP	Proposto
crud	FUI	Exclui {objeto} Web FUI Java/JSP/EJB	Publicado
crud	FUI	Exclui {objeto} Web FUI PHP	Publicado
crud	SUI	Exclui {objeto} Desktop SUI	Publicado
crud	CUI	Exclui {objeto} Desktop CUI Básica	Proposto
crud	FUI	Exclui {objeto} Desktop FUI Java/Swing	Publicado
crud	FUI	Exclui {objeto} Desktop FUI VB	Publicado
crud	CUCP	Mantém {informacao}	Publicado
crud	SUI	Mantém {informacao} Web SUI	Publicado
crud	CUI	Mantém {informacao} Web CUI Básica	Proposto
crud	FUI	Mantém {informacao} Web FUI XHTML/CSS	Publicado
crud	FUI	Mantém {informacao} Web FUI VB/ASP	Publicado
crud	FUI	Mantém {informacao} Web FUI Java/JSP	Proposto
crud	FUI	Mantém {informacao} Web FUI Java/JSP/EJB	Publicado
crud	FUI	Mantém {informacao} Web FUI PHP	Publicado
crud	SUI	Mantém {informacao} Desktop SUI	Publicado
crud	CUI	Mantém {informacao} Desktop CUI Básica	Proposto
crud	FUI	Mantém {informacao} Desktop FUI Java/Swing	Publicado
crud	FUI	Mantém {informacao} Desktop FUI VB	Publicado

Pacote	Tipo	Asset	Situação
crud	SUI	Mantém {informacao} ordem manual Desktop SUI	Publicado
crud	CUI	Mantém {informacao} ordem manual Desktop CUI Básica	Proposto
crud	FUI	Mantém {informacao} ordem manual Desktop FUI Java/Swing	Publicado
crud	FUI	Mantém {informacao} ordem manual Desktop FUI VB	Publicado
crud	CUCP	Imprime {objeto}	Publicado
crud	SUI	Imprime {objeto} Web SUI	Publicado
crud	CUI	Imprime {objeto} Web CUI Básica	Proposto
crud	FUI	Imprime {objeto} Web FUI XHTML/CSS	Publicado
crud	FUI	Imprime {objeto} Web FUI VB/ASP	Publicado
crud	FUI	Imprime {objeto} Web FUI Java/JSP	Proposto
crud	FUI	Imprime {objeto} Web FUI Java/JSP/EJB	Publicado
crud	FUI	Imprime {objeto} Web FUI PHP	Publicado
crud	SUI	Imprime {objeto} Desktop SUI	Publicado
crud	CUI	Imprime {objeto} Desktop CUI Básica	Proposto
crud	FUI	Imprime {objeto} Desktop FUI Java/Swing	Publicado
crud	FUI	Imprime {objeto} Desktop FUI VB	Publicado
crud	CUCP	Imprime lista de {objeto}	Publicado
crud	SUI	Imprime lista de {objeto} Web SUI	Publicado
crud	CUI	Imprime lista de {objeto} Web CUI Básica	Proposto
crud	FUI	Imprime lista de {objeto} Web FUI XHTML/CSS	Publicado
crud	FUI	Imprime lista de {objeto} Web FUI VB/ASP	Publicado
crud	FUI	Imprime lista de {objeto} Web FUI Java/JSP	Proposto
crud	FUI	Imprime lista de {objeto} Web FUI Java/JSP/EJB	Publicado
crud	FUI	Imprime lista de {objeto} Web FUI PHP	Publicado
crud	SUI	Imprime lista de {objeto} Desktop SUI	Publicado
crud	CUI	Imprime lista de {objeto} Desktop CUI Básica	Proposto
crud	FUI	Imprime lista de {objeto} Desktop FUI Java/Swing	Publicado
crud	FUI	Imprime lista de {objeto} Desktop FUI VB	Publicado
crud	CUCP	Visualiza impressão	Publicado
crud	SUI	Visualiza impressão Desktop SUI	Publicado
crud	CUI	Visualiza impressão Desktop CUI Básica	Proposto
crud	FUI	Visualiza impressão Desktop FUI Java/Swing	Publicado
crud	FUI	Visualiza impressão Desktop FUI VB	Publicado
accesscontrol	CUCP	Identificar-se para o sistema (Login)	Publicado
accesscontrol	AUI	Identificar-se para o sistema (Login) AUI	Publicado
accesscontrol	SUI	Identificar-se para o sistema (Login) Web SUI	Publicado
accesscontrol	XUI	Identificar-se para o sistema (Login) Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Identificar-se para o sistema (Login) Web XUI VB/ASP	Publicado
accesscontrol	SUI	Identificar-se para o sistema (Login) Desktop SUI	Publicado

Pacote	Tipo	Asset	Situação
accesscontrol	XUI	Identificar-se para o sistema (Login) Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Identificar-se para o sistema (Login) Desktop XUI VB	Publicado
accesscontrol	CUCP	Encerra sessão	Publicado
accesscontrol	AUI	Encerra sessão AUI	Publicado
accesscontrol	SUI	Encerra sessão Web SUI	Publicado
accesscontrol	XUI	Encerra sessão Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Encerra sessão Web XUI VB/ASP	Publicado
accesscontrol	SUI	Encerra sessão Desktop SUI	Publicado
accesscontrol	XUI	Encerra sessão Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Encerra sessão Desktop XUI VB	Publicado
accesscontrol	CUCP	Altera senha	Publicado
accesscontrol	AUI	Altera senha AUI	Publicado
accesscontrol	SUI	Altera senha Web SUI	Publicado
accesscontrol	XUI	Altera senha Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Altera senha Web XUI VB/ASP	Publicado
accesscontrol	SUI	Altera senha Desktop SUI	Publicado
accesscontrol	XUI	Altera senha Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Altera senha Desktop XUI VB	Publicado
accesscontrol	CUCP	Solicita nova senha	Publicado
accesscontrol	AUI	Solicita nova senha AUI	Publicado
accesscontrol	SUI	Solicita nova senha Web SUI	Publicado
accesscontrol	XUI	Solicita nova senha Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Solicita nova senha Web XUI VB/ASP	Publicado
accesscontrol	SUI	Solicita nova senha Desktop SUI	Publicado
accesscontrol	XUI	Solicita nova senha Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Solicita nova senha Desktop XUI VB	Publicado
accesscontrol	CUCP	Inicializa senha do usuário	Publicado
accesscontrol	AUI	Inicializa senha do usuário AUI	Publicado
accesscontrol	SUI	Inicializa senha do usuário Web SUI	Publicado
accesscontrol	XUI	Inicializa senha do usuário Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Inicializa senha do usuário Web XUI VB/ASP	Publicado
accesscontrol	SUI	Inicializa senha do usuário Desktop SUI	Publicado
accesscontrol	XUI	Inicializa senha do usuário Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Inicializa senha do usuário Desktop XUI VB	Publicado
accesscontrol	CUCP	Pesquisa usuário	Publicado
accesscontrol	AUI	Pesquisa usuário AUI	Publicado
accesscontrol	SUI	Pesquisa usuário Web SUI	Publicado
accesscontrol	XUI	Pesquisa usuário Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Pesquisa usuário Web XUI VB/ASP	Publicado
accesscontrol	SUI	Pesquisa usuário Desktop SUI	Publicado

Pacote	Tipo	Asset	Situação
accesscontrol	XUI	Pesquisa usuário Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Pesquisa usuário Desktop XUI VB	Publicado
accesscontrol	CUCP	Lista usuário	Publicado
accesscontrol	AUI	Lista usuário AUI	Publicado
accesscontrol	SUI	Lista usuário Web SUI	Publicado
accesscontrol	XUI	Lista usuário Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Lista usuário Web XUI VB/ASP	Publicado
accesscontrol	SUI	Lista usuário Desktop SUI	Publicado
accesscontrol	XUI	Lista usuário Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Lista usuário Desktop XUI VB	Publicado
accesscontrol	CUCP	Inclui usuário	Publicado
accesscontrol	AUI	Inclui usuário AUI	Publicado
accesscontrol	SUI	Inclui usuário Web SUI	Publicado
accesscontrol	XUI	Inclui usuário Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Inclui usuário Web XUI VB/ASP	Publicado
accesscontrol	SUI	Inclui usuário Desktop SUI	Publicado
accesscontrol	XUI	Inclui usuário Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Inclui usuário Desktop XUI VB	Publicado
accesscontrol	CUCP	Altera usuário	Publicado
accesscontrol	AUI	Altera usuário AUI	Publicado
accesscontrol	SUI	Altera usuário Web SUI	Publicado
accesscontrol	XUI	Altera usuário Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Altera usuário Web XUI VB/ASP	Publicado
accesscontrol	SUI	Altera usuário Desktop SUI	Publicado
accesscontrol	XUI	Altera usuário Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Altera usuário Desktop XUI VB	Publicado
accesscontrol	CUCP	Exclui usuário	Publicado
accesscontrol	AUI	Exclui usuário AUI	Publicado
accesscontrol	SUI	Exclui usuário Web SUI	Publicado
accesscontrol	XUI	Exclui usuário Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Exclui usuário Web XUI VB/ASP	Publicado
accesscontrol	SUI	Exclui usuário Desktop SUI	Publicado
accesscontrol	XUI	Exclui usuário Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Exclui usuário Desktop XUI VB	Publicado
accesscontrol	CUCP	Mantém nível de permissão	Publicado
accesscontrol	AUI	Mantém nível de permissão AUI	Publicado
accesscontrol	SUI	Mantém nível de permissão Web SUI	Publicado
accesscontrol	XUI	Mantém nível de permissão Web XUI Java/JSP/EJB	Publicado
accesscontrol	XUI	Mantém nível de permissão Web XUI VB/ASP	Publicado
accesscontrol	SUI	Mantém nível de permissão Desktop SUI	Publicado

Pacote	Tipo	Asset	Situação
accesscontrol	XUI	Mantém nível de permissão Desktop XUI Java/Swing	Publicado
accesscontrol	XUI	Mantém nível de permissão Desktop XUI VB	Publicado
noticias	CUCP	Pesquisa notícia	Proposto
noticias	AUI	Pesquisa notícia AUI	Proposto
noticias	SUI	Pesquisa notícia Web SUI	Proposto
noticias	CUI	Pesquisa notícia Web CUI Básica	Proposto
noticias	FUI	Pesquisa notícia Web FUI PHP	Proposto
noticias	CUCP	Lista notícia	Proposto
noticias	AUI	Lista notícia AUI	Proposto
noticias	SUI	Lista notícia Web SUI	Proposto
noticias	CUI	Lista notícia Web CUI Básica	Proposto
noticias	FUI	Lista notícia Web FUI PHP	Proposto
noticias	CUCP	Visualiza notícia	Proposto
noticias	AUI	Visualiza notícia AUI	Proposto
noticias	SUI	Visualiza notícia Web SUI	Proposto
noticias	CUI	Visualiza notícia Web CUI Básica	Proposto
noticias	FUI	Visualiza notícia Web FUI PHP	Proposto
noticias	CUCP	Inclui notícia	Proposto
noticias	AUI	Inclui notícia AUI	Proposto
noticias	SUI	Inclui notícia Web SUI	Proposto
noticias	CUI	Inclui notícia Web CUI Básica	Proposto
noticias	FUI	Inclui notícia Web FUI PHP	Proposto
noticias	CUCP	Altera notícia	Proposto
noticias	AUI	Altera notícia AUI	Proposto
noticias	SUI	Altera notícia Web SUI	Proposto
noticias	CUI	Altera notícia Web CUI Básica	Proposto
noticias	FUI	Altera notícia Web FUI PHP	Proposto
noticias	CUCP	Exclui notícia	Proposto
noticias	AUI	Exclui notícia AUI	Proposto
noticias	SUI	Exclui notícia Web SUI	Proposto
noticias	CUI	Exclui notícia Web CUI Básica	Proposto
noticias	FUI	Exclui notícia Web FUI PHP	Proposto
noticias	CUCP	Imprime notícia	Proposto
noticias	AUI	Imprime notícia AUI	Proposto
noticias	SUI	Imprime notícia Web SUI	Proposto
noticias	CUI	Imprime notícia Web CUI Básica	Proposto
noticias	FUI	Imprime notícia Web FUI PHP	Proposto
noticias	CUCP	Publica notícia	Proposto
noticias	AUI	Publica notícia AUI	Proposto
noticias	SUI	Publica notícia Web SUI	Proposto

Pacote	Tipo	Asset	Situação
noticias	CUI	Publica notícia Web CUI Básica	Proposto
noticias	FUI	Publica notícia Web FUI PHP	Proposto
noticias	CUCP	Cancela publicação de notícia	Proposto
noticias	AUI	Cancela publicação de notícia AUI	Proposto
noticias	SUI	Cancela publicação de notícia Web SUI	Proposto
noticias	CUI	Cancela publicação de notícia Web CUI Básica	Proposto
noticias	FUI	Cancela publicação de notícia Web FUI PHP	Proposto
noticias	CUCP	Mantém categoria de notícia	Proposto
noticias	AUI	Mantém categoria de notícia AUI	Proposto
noticias	SUI	Mantém categoria de notícia Web SUI	Proposto
noticias	CUI	Mantém categoria de notícia Web CUI Básica	Proposto
noticias	FUI	Mantém categoria de notícia Web FUI PHP	Proposto
links	CUCP	Lista link	Proposto
links	AUI	Lista link AUI	Proposto
links	SUI	Lista link Web SUI	Proposto
links	CUI	Lista link Web CUI Básica	Proposto
links	FUI	Lista link Web FUI PHP	Proposto
links	CUCP	Inclui link	Proposto
links	AUI	Inclui link AUI	Proposto
links	SUI	Inclui link Web SUI	Proposto
links	CUI	Inclui link Web CUI Básica	Proposto
links	FUI	Inclui link Web FUI PHP	Proposto
links	CUCP	Altera link	Proposto
links	AUI	Altera link AUI	Proposto
links	SUI	Altera link Web SUI	Proposto
links	CUI	Altera link Web CUI Básica	Proposto
links	FUI	Altera link Web FUI PHP	Proposto
links	CUCP	Exclui link	Proposto
links	AUI	Exclui link AUI	Proposto
links	SUI	Exclui link Web SUI	Proposto
links	CUI	Exclui link Web CUI Básica	Proposto
links	FUI	Exclui link Web FUI PHP	Proposto
links	CUCP	Mantém categoria de link	Proposto
links	AUI	Mantém categoria de link AUI	Proposto
links	SUI	Mantém categoria de link Web SUI	Proposto
links	CUI	Mantém categoria de link Web CUI Básica	Proposto
links	FUI	Mantém categoria de link Web FUI PHP	Proposto
ajuda	CUCP	Solicita ajuda de uma tarefa	Proposto
ajuda	AUI	Solicita ajuda de uma tarefa AUI	Proposto
ajuda	SUI	Solicita ajuda de uma tarefa Web SUI	Proposto

Pacote	Tipo	Asset	Situação
ajuda	CUI	Solicita ajuda de uma tarefa Web CUI Básica	Proposto
ajuda	CUCP	Solicita ajuda contextual	Proposto
ajuda	AUI	Solicita ajuda contextual AUI	Proposto
ajuda	SUI	Solicita ajuda contextual Web SUI	Proposto
ajuda	CUI	Solicita ajuda contextual Web CUI Básica	Proposto

APÊNDICE D – PADRÕES CONCRETOS DE INTERAÇÃO DO ESTUDO DE CASO

A tabela a seguir apresenta os assets dos padrões concretos de interação que foram catalogados e utilizados no estudo de caso (assets com a situação: “Publicado”). A tabela também relaciona os assets que foram identificados ao longo do processo de desenvolvimento do sistema como sendo candidatos a serem construídos para futuro reuso (assets com a situação: “Proposto”).

Tabela D.1: Padrões concretos de interação do estudo de caso.

Coleção / Categoria	Tipo	Asset	Situação
Augusto Abelin Moreira			
Edição de dados primitivos GUI			
	CIP	Edição de texto de uma linha	Publicado
	SUI	Edição de texto de uma linha SUI	Publicado
	CUI	Edição de texto de uma linha CUI Genérica Completa	Proposto
	XUI	Edição de texto de uma linha XUI VB	Publicado
	XUI	Edição de texto de uma linha XUI Java/Swing	Publicado
	CIP	Edição de texto com histórico de strings	Publicado
	SUI	Edição de texto com histórico de strings SUI	Publicado
	CUI	Edição de texto com histórico de strings CUI Genérica Completa	Proposto
	XUI	Edição de texto com histórico de strings XUI VB	Publicado
	XUI	Edição de texto com histórico de strings XUI Java/Swing	Publicado
	CIP	Edição de números inteiros	Publicado
	SUI	Edição de números inteiros SUI	Publicado
	CUI	Edição de números inteiros CUI Genérica Completa	Proposto
	XUI	Edição de números inteiros XUI VB	Publicado
	XUI	Edição de números inteiros XUI Java/Swing	Publicado
	CIP	Edição de números fracionários	Publicado
	SUI	Edição de números fracionários SUI	Publicado
	CUI	Edição de números fracionários CUI Genérica Completa	Proposto
	XUI	Edição de números fracionários XUI VB	Publicado
	XUI	Edição de números fracionários XUI Java/Swing	Publicado

Coleção / Categoria	Tipo	Asset	Situação
	CIP	Edição de data	Publicado
	CIP	Edição de data estruturada	Publicado
	SUI	Edição de data estruturada SUI	Publicado
	CUI	Edição de data estruturada CUI Genérica Completa	Proposto
	XUI	Edição de data estruturada XUI VB	Publicado
	XUI	Edição de data estruturada XUI Java/Swing	Publicado
	CIP	Calendário	Publicado
	SUI	Calendário SUI	Publicado
	CUI	Calendário CUI Genérica Completa	Proposto
	XUI	Calendário XUI VB	Publicado
	XUI	Calendário XUI Java/Swing	Publicado
	CIP	Edição de texto multi-linha sem formatação	Publicado
	SUI	Edição de texto multi-linha sem formatação SUI	Publicado
	CUI	Edição de texto multi-linha sem formatação CUI Genérica Completa	Proposto
	XUI	Edição de texto multi-linha sem formatação XUI VB	Publicado
	XUI	Edição de texto multi-linha sem formatação XUI Java/Swing	Publicado
	CIP	Edição de texto multi-linha com formatação	Publicado
	SUI	Edição de texto multi-linha com formatação SUI	Publicado
	CUI	Edição de texto multi-linha com formatação CUI Genérica Completa	Proposto
	XUI	Edição de texto multi-linha com formatação XUI VB	Publicado
	XUI	Edição de texto multi-linha com formatação XUI Java/Swing	Publicado
	CIP	Ordenação manual de lista	Publicado
	SUI	Ordenação manual de lista SUI	Publicado
	CUI	Ordenação manual de lista CUI Genérica Completa	Proposto
	XUI	Ordenação manual de lista XUI VB	Publicado
	XUI	Ordenação manual de lista XUI Java/Swing	Publicado
Elementos de domínio primitivo GUI			
	CIP	Edição de número de telefone brasileiro	Publicado
	SUI	Edição de número de telefone brasileiro SUI	Publicado
	CUI	Edição de número de telefone brasileiro CUI Genérica Completa	Proposto
	XUI	Edição de número de telefone brasileiro XUI VB	Publicado
	XUI	Edição de número de telefone brasileiro XUI Java/Swing	Publicado
	CIP	Edição de lista de telefones	Publicado
	SUI	Edição de lista de telefones SUI	Publicado
	CUI	Edição de lista de telefones CUI Genérica Completa	Proposto
	FUI	Edição lista de telefones FUI VB	Publicado
	FUI	Edição de lista de telefones FUI Java/Swing	Publicado
	CIP	Edição de e-mail	Publicado

Coleção / Categoria	Tipo	Asset	Situação
	SUI	Edição de e-mail SUI	Publicado
	CUI	Edição de e-mail CUI Genérica Completa	Proposto
	XUI	Edição de e-mail XUI VB	Publicado
	XUI	Edição de e-mail XUI Java/Swing	Publicado
	CIP	Edição de lista de e-mails	Publicado
	SUI	Edição de lista de e-mails SUI	Publicado
	CUI	Edição de lista de e-mails CUI Genérica Completa	Proposto
	FUI	Edição de lista de e-mails FUI VB	Publicado
	FUI	Edição de lista de e-mails FUI Java/Swing	Publicado
	CIP	Edição de logradouro	Publicado
	SUI	Edição de logradouro SUI	Publicado
	CUI	Edição de logradouro CUI Genérica Completa	Proposto
	FUI	Edição de logradouro FUI VB	Publicado
	FUI	Edição de logradouro FUI Java/Swing	Publicado
	CIP	Edição de endereço brasileiro simples	Publicado
	SUI	Edição de endereço brasileiro simples SUI	Publicado
	CUI	Edição de endereço brasileiro simples CUI Genérica Completa	Proposto
	FUI	Edição de endereço brasileiro simples FUI VB	Publicado
	FUI	Edição de endereço brasileiro simples FUI Java/Swing	Publicado
	CIP	Edição de endereço brasileiro estruturado 1 (UF/Município)	Publicado
	SUI	Edição de endereço brasileiro estruturado 1 (UF/Município)SUI	Publicado
	CUI	Edição de endereço brasileiro estruturado 1 (UF/Município) CUI Genérica Completa	Proposto
	FUI	Edição de endereço brasileiro estruturado 1 (UF/Município) FUI VB	Publicado
	FUI	Edição de endereço brasileiro estruturado 1 (UF/Município) FUI Java/Swing	Publicado
	CIP	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro)	Publicado
	SUI	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) SUI	Publicado
	CUI	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) CUI Genérica Completa	Proposto
	FUI	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) FUI VB	Publicado
	FUI	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) FUI Java/Swing	Publicado
	CIP	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro)	Publicado
	SUI	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) SUI	Publicado
	CUI	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) CUI Genérica Completa	Proposto
	FUI	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) FUI VB	Publicado
	FUI	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) FUI Java/Swing	Publicado
	CIP	Edição de CPF	Publicado

Coleção / Categoria	Tipo	Asset	Situação
	SUI	Edição de CPF SUI	Publicado
	CUI	Edição de CPF CUI Genérica Completa	Proposto
	XUI	Edição de CPF XUI VB	Publicado
	XUI	Edição de CPF XUI Java/Swing	Publicado
	CIP	Edição de CNPJ	Publicado
	SUI	Edição de CNPJ SUI	Publicado
	CUI	Edição de CNPJ CUI Genérica Completa	Proposto
	XUI	Edição de CNPJ XUI VB	Publicado
	XUI	Edição de CNPJ XUI Java/Swing	Publicado
Aplicação desktop			
	CIP	Aplicativo MDI	Publicado
	SUI	Aplicativo MDI SUI	Publicado
	CUI	Aplicativo MDI CUI Genérica Completa	Proposto
	FUI	Aplicativo MDI FUI VB	Publicado
	FUI	Aplicativo MDI FUI Java/Swing	Publicado
	CIP	Splash	Publicado
	SUI	Splash SUI	Publicado
	CUI	Splash CUI Genérica Completa	Proposto
	FUI	Splash FUI VB	Publicado
	FUI	Splash FUI Java/Swing	Publicado
	CIP	Assistente	Publicado
	SUI	Assistente SUI	Publicado
	CUI	Assistente CUI Genérica Completa	Proposto
	FUI	Assistente FUI VB	Publicado
	FUI	Assistente FUI Java/Swing	Publicado
	CIP	Sobre	Publicado
	SUI	Sobre SUI	Publicado
	CUI	Sobre CUI Genérica Completa	Proposto
	FUI	Sobre FUI VB	Publicado
	FUI	Sobre FUI Java/Swing	Publicado
Pesquisa de bases de dados GUI			
	CIP	Pesquisa simples	Publicado
	SUI	Pesquisa simples SUI	Publicado
	CUI	Pesquisa simples CUI Genérica Completa	Proposto
	FUI	Pesquisa simples FUI VB	Publicado
	FUI	Pesquisa simples FUI Java/Swing	Publicado
	CIP	Pesquisa avançada	Publicado
	SUI	Pesquisa avançada SUI	Publicado

Coleção / Categoria	Tipo	Asset	Situação
	CUI	Pesquisa avançada CUI Genérica Completa	Proposto
	FUI	Pesquisa avançada FUI VB	Publicado
	FUI	Pesquisa avançada FUI Java/Swing	Publicado
	CIP	Pesquisa avançada com salvamento de critérios de pesquisa	Publicado
	SUI	Pesquisa avançada com salvamento de critérios de pesquisa SUI	Publicado
	CUI	Pesquisa avançada com salvamento de critérios de pesquisa CUI Genérica Completa	Proposto
	FUI	Pesquisa avançada com salvamento de critérios de pesquisa FUI VB	Publicado
	FUI	Pesquisa avançada com salvamento de critérios de pesquisa FUI Java/Swing	Publicado
	CIP	Lista	Publicado
	SUI	Lista SUI	Publicado
	CUI	Lista CUI Genérica Completa	Proposto
	FUI	Lista FUI VB	Publicado
	FUI	Lista FUI Java/Swing	Publicado
Manutenção de bases de dados GUI			
	CIP	Inclusão única	Publicado
	SUI	Inclusão única SUI	Publicado
	CUI	Inclusão única CUI Genérica Completa	Proposto
	FUI	Inclusão única FUI VB	Publicado
	FUI	Inclusão única FUI Java/Swing	Publicado
	CIP	Inclusão múltipla	Publicado
	SUI	Inclusão múltipla SUI	Publicado
	CUI	Inclusão múltipla CUI Genérica Completa	Proposto
	FUI	Inclusão múltipla FUI VB	Publicado
	FUI	Inclusão múltipla FUI Java/Swing	Publicado
	CIP	Editar	Publicado
	SUI	Editar SUI	Publicado
	CUI	Editar CUI Genérica Completa	Proposto
	FUI	Editar FUI VB	Publicado
	FUI	Editar FUI Java/Swing	Publicado
	CIP	Associação simples com lista reduzida	Publicado
	SUI	Associação simples com lista reduzida SUI	Publicado
	CUI	Associação simples com lista reduzida CUI Genérica Completa	Proposto
	FUI	Associação simples com lista reduzida FUI VB	Publicado
	FUI	Associação simples com lista reduzida FUI Java/Swing	Publicado
	CIP	Associação simples com lista extensa	Publicado
	SUI	Associação simples com lista extensa SUI	Publicado
	CUI	Associação simples com lista extensa CUI Genérica Completa	Proposto

Coleção / Categoria	Tipo	Asset	Situação
	FUI	Associação simples com lista extensa FUI VB	Publicado
	FUI	Associação simples com lista extensa FUI Java/Swing	Publicado
	CIP	Associação dupla	Publicado
	SUI	Associação dupla SUI	Publicado
	CUI	Associação dupla CUI Genérica Completa	Proposto
	FUI	Associação dupla FUI VB	Publicado
	FUI	Associação dupla FUI Java/Swing	Publicado
	CIP	Associação dupla com ordenação manual	Publicado
	SUI	Associação dupla com ordenação manual SUI	Publicado
	CUI	Associação dupla com ordenação manual CUI Genérica Completa	Proposto
	FUI	Associação dupla com ordenação manual FUI VB	Publicado
	FUI	Associação dupla com ordenação manual FUI Java/Swing	Publicado
	CIP	Associação dupla de hierarquias	Publicado
	SUI	Associação dupla de hierarquias SUI	Publicado
	CUI	Associação dupla de hierarquias CUI Genérica Completa	Proposto
	FUI	Associação dupla de hierarquias FUI VB	Publicado
	FUI	Associação dupla de hierarquias FUI Java/Swing	Proposto
	CIP	Manutenção de tabela simples ordenada	Publicado
	SUI	Manutenção de tabela simples ordenada SUI	Publicado
	CUI	Manutenção de tabela simples ordenada CUI Genérica Completa	Proposto
	FUI	Manutenção de tabela simples ordenada FUI VB	Publicado
	FUI	Manutenção de tabela simples ordenada FUI Java/Swing	Publicado
	CIP	Manutenção de tabela simples com ordenação manual	Publicado
	SUI	Manutenção de tabela simples com ordenação manual SUI	Publicado
	CUI	Manutenção de tabela simples com ordenação manual CUI Genérica Completa	Proposto
	FUI	Manutenção de tabela simples com ordenação manual FUI VB	Publicado
	FUI	Manutenção de tabela simples com ordenação manual FUI Java/Swing	Publicado
Diálogos padrões			
	CIP	Abrir arquivo	Publicado
	SUI	Abrir arquivo SUI	Publicado
	CUI	Abrir arquivo CUI Genérica Completa	Proposto
	XUI	Abrir arquivo XUI VB	Publicado
	XUI	Abrir arquivo XUI Java/Swing	Publicado
	CIP	Salvar arquivo	Publicado
	SUI	Salvar arquivo SUI	Publicado
	CUI	Salvar arquivo CUI Genérica Completa	Proposto
	XUI	Salvar arquivo XUI VB	Publicado
	XUI	Salvar arquivo XUI Java/Swing	Publicado

Coleção / Categoria	Tipo	Asset	Situação
	CIP	Imprimir	Publicado
	SUI	Imprimir SUI	Publicado
	CUI	Imprimir CUI Genérica Completa	Proposto
	XUI	Imprimir XUI VB	Publicado
	XUI	Imprimir XUI Java/Swing	Publicado
	CIP	Visualizar impressão	Publicado
	SUI	Visualizar impressão SUI	Publicado
	CUI	Visualizar impressão CUI Genérica Completa	Proposto
	XUI	Visualizar impressão XUI VB	Publicado
	XUI	Visualizar impressão XUI Java/Swing	Publicado
	CIP	Configurar impressora	Publicado
	SUI	Configurar impressora SUI	Publicado
	CUI	Configurar impressora CUI Genérica Completa	Proposto
	XUI	Configurar impressora XUI VB	Publicado
	XUI	Configurar impressora XUI Java/Swing	Publicado
	CIP	Mensagem informativa	Publicado
	SUI	Mensagem informativa SUI	Publicado
	CUI	Mensagem informativa CUI Genérica Completa	Proposto
	XUI	Mensagem informativa XUI VB	Publicado
	XUI	Mensagem informativa XUI Java/Swing	Publicado
	CIP	Mensagem de erro	Publicado
	SUI	Mensagem de erro SUI	Publicado
	CUI	Mensagem de erro CUI Genérica Completa	Proposto
	XUI	Mensagem de erro XUI VB	Publicado
	XUI	Mensagem de erro XUI Java/Swing	Publicado
	CIP	Pergunta com respostas simples	Publicado
	SUI	Pergunta com respostas simples SUI	Publicado
	CUI	Pergunta com respostas simples CUI Genérica Completa	Proposto
	XUI	Pergunta com respostas simples XUI VB	Publicado
	XUI	Pergunta com respostas simples XUI Java/Swing	Publicado
	CIP	Pergunta com respostas complexas	Publicado
	SUI	Pergunta com respostas complexas SUI	Publicado
	CUI	Pergunta com respostas complexas CUI Genérica Completa	Proposto
	XUI	Pergunta com respostas complexas XUI VB	Publicado
	XUI	Pergunta com respostas complexas XUI Java/Swing	Publicado
Edição de dados primitivos Web			
	CIP	Edição de números inteiros Web	Publicado
	SUI	Edição de números inteiros Web SUI	Publicado

Coleção / Categoria	Tipo	Asset	Situação
	CUI	Edição de números inteiros Web CUI Genérica Completa	Proposto
	FUI	Edição de números inteiros Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de números fracionários Web	Publicado
	SUI	Edição de números fracionários Web SUI	Publicado
	CUI	Edição de números fracionários Web CUI Genérica Completa	Proposto
	FUI	Edição de números fracionários Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de data Web	Publicado
	SUI	Edição de data Web SUI	Publicado
	CUI	Edição de data Web CUI Genérica Completa	Proposto
	FUI	Edição de data Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de data estruturada Web	Publicado
	SUI	Edição de data estruturada Web SUI	Publicado
	CUI	Edição de data estruturada Web CUI Genérica Completa	Proposto
	FUI	Edição de data estruturada Web FUI XHTML/JavaScript	Publicado
	CIP	Calendário Web	Publicado
	SUI	Calendário Web SUI	Publicado
	CUI	Calendário Web CUI Genérica Completa	Proposto
	FUI	Calendário Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de hora Web	Publicado
	SUI	Edição de hora Web SUI	Publicado
	CUI	Edição de hora Web CUI Genérica Completa	Proposto
	FUI	Edição de hora Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de texto multi-linha com formatação Web	Publicado
	SUI	Edição de texto multi-linha com formatação Web SUI	Publicado
	CUI	Edição de texto multi-linha com formatação Web CUI Genérica Completa	Proposto
	FUI	Edição de texto multi-linha com formatação Web FUI XHTML/JavaScript	Publicado
	CIP	Ordenação manual de lista Web	Publicado
	SUI	Ordenação manual de lista Web SUI	Publicado
	CUI	Ordenação manual de lista Web CUI Genérica Completa	Proposto
	FUI	Ordenação manual de lista Web FUI XHTML/JavaScript	Publicado
Elementos de domínio primitivo Web			
	CIP	Edição de número de telefone brasileiro Web	Publicado
	SUI	Edição de número de telefone brasileiro Web SUI	Publicado
	CUI	Edição de número de telefone brasileiro Web CUI Genérica Completa	Proposto
	FUI	Edição de número de telefone brasileiro Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de lista de telefones Web	Publicado
	SUI	Edição de lista de telefones Web SUI	Publicado
	CUI	Edição de lista de telefones Web CUI Genérica Completa	Proposto

Coleção / Categoria	Tipo	Asset	Situação
	FUI	Edição de lista de telefones Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de e-mail Web	Publicado
	SUI	Edição de e-mail Web SUI	Publicado
	CUI	Edição de e-mail Web CUI Genérica Completa	Proposto
	FUI	Edição de e-mail Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de lista de e-mails Web	Publicado
	SUI	Edição de lista de e-mails Web SUI	Publicado
	CUI	Edição de lista de e-mails Web CUI Genérica Completa	Proposto
	FUI	Edição de lista de e-mails Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de logradouro Web	Publicado
	SUI	Edição de logradouro Web SUI	Publicado
	CUI	Edição de logradouro Web CUI Genérica Completa	Proposto
	FUI	Edição de logradouro Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de endereço brasileiro simples Web	Publicado
	SUI	Edição de endereço brasileiro simples Web SUI	Publicado
	CUI	Edição de endereço brasileiro simples Web CUI Genérica Completa	Proposto
	FUI	Edição de endereço brasileiro simples Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de endereço brasileiro estruturado 1 (UF/Município) Web	Publicado
	SUI	Edição de endereço brasileiro estruturado 1 (UF/Município) Web SUI	Publicado
	CUI	Edição de endereço brasileiro estruturado 1 (UF/Município) Web CUI Genérica Completa	Proposto
	FUI	Edição de endereço brasileiro estruturado 1 (UF/Município) Web FUI XHTML/JavaScript	Publicado
	FUI	Edição de endereço brasileiro estruturado 1 (UF/Município) Web FUI Java/JSP	Proposto
	FUI	Edição de endereço brasileiro estruturado 1 (UF/Município) Web FUI VB/ASP	Proposto
	FUI	Edição de endereço brasileiro estruturado 1 (UF/Município) Web FUI PHP	Proposto
	CIP	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) Web	Publicado
	SUI	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) Web SUI	Publicado
	CUI	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) Web CUI Genérica Completa	Proposto
	FUI	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) Web FUI XHTML/JavaScript	Publicado
	FUI	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) Web FUI Java/JSP	Proposto
	FUI	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) Web FUI VB/ASP	Proposto
	FUI	Edição de endereço brasileiro estruturado 2 (UF/Município/Bairro) Web FUI PHP	Proposto
	CIP	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) Web	Publicado
	SUI	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) Web SUI	Publicado
	CUI	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) Web CUI Genérica Completa	Proposto

Coleção / Categoria	Tipo	Asset	Situação
	FUI	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) Web FUI XHTML/JavaScript	Publicado
	FUI	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) Web FUI Java/JSP	Proposto
	FUI	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) Web FUI VB/ASP	Proposto
	FUI	Edição de endereço brasileiro estruturado 3 (CEP/UF/Município/Bairro) Web FUI PHP	Proposto
	CIP	Edição de CPF Web	Publicado
	SUI	Edição de CPF Web SUI	Publicado
	CUI	Edição de CPF Web CUI Genérica Completa	Proposto
	FUI	Edição de CPF Web FUI XHTML/JavaScript	Publicado
	CIP	Edição de CNPJ Web	Publicado
	SUI	Edição de CNPJ Web SUI	Publicado
	CUI	Edição de CNPJ Web CUI Genérica Completa	Proposto
	FUI	Edição de CNPJ Web FUI XHTML/JavaScript	Publicado
Aplicação Web			
	CIP	Aplicação Web	Publicado
	SUI	Aplicação Web SUI	Publicado
	CUI	Aplicação Web CUI Genérica Completa	Proposto
	FUI	Aplicação Web FUI Java/JSP	Publicado
	FUI	Aplicação Web FUI VB/ASP	Publicado
	FUI	Aplicação Web FUI PHP	Publicado
	CIP	Assistente Web	Publicado
	SUI	Assistente Web SUI	Publicado
	CUI	Assistente Web CUI Genérica Completa	Proposto
	FUI	Assistente Web FUI XHTML/JavaScript	Publicado
Pesquisa de bases de dados Web			
	CIP	Busca	Publicado
	SUI	Busca Web SUI	Publicado
	CUI	Busca Web CUI Genérica Completa	Proposto
	FUI	Busca Web FUI XHTML/JavaScript	Publicado
	FUI	Busca Web FUI Java/JSP	Publicado
	FUI	Busca Web FUI VB/ASP	Publicado
	FUI	Busca Web FUI PHP	Publicado
	CIP	Busca avançada Web	Publicado
	SUI	Busca avançada Web SUI	Publicado
	CUI	Busca avançada Web CUI Genérica Completa	Proposto
	FUI	Busca avançada Web FUI XHTML/JavaScript	Publicado
	FUI	Busca avançada Web FUI Java/JSP	Publicado

Coleção / Categoria	Tipo	Asset	Situação
	FUI	Busca avançada Web FUI VB/ASP	Publicado
	FUI	Busca avançada Web FUI PHP	Publicado
	CIP	Lista sem paginação	Publicado
	SUI	Lista sem paginação SUI	Publicado
	CUI	Lista sem paginação CUI Genérica Completa	Proposto
	FUI	Lista sem paginação FUI XHTML/JavaScript	Publicado
	FUI	Lista sem paginação FUI Java/JSP	Publicado
	FUI	Lista sem paginação FUI VB/ASP	Publicado
	FUI	Lista sem paginação FUI PHP	Publicado
	CIP	Lista com paginação	Publicado
	SUI	Lista com paginação SUI	Publicado
	CUI	Lista com paginação CUI Genérica Completa	Proposto
	FUI	Lista com paginação FUI XHTML/JavaScript	Publicado
	FUI	Lista com paginação FUI Java/JSP	Publicado
	FUI	Lista com paginação FUI VB/ASP	Publicado
	FUI	Lista com paginação FUI PHP	Publicado
Manutenção de bases de dados Web			
	CIP	Inclusão única	Publicado
	SUI	Inclusão única SUI	Publicado
	CUI	Inclusão única CUI Genérica Completa	Proposto
	FUI	Inclusão única FUI XHTML/JavaScript	Publicado
	FUI	Inclusão única FUI Java/JSP	Publicado
	FUI	Inclusão única FUI VB/ASP	Publicado
	FUI	Inclusão única FUI PHP	Publicado
	CIP	Inclusão múltipla	Publicado
	SUI	Inclusão múltipla SUI	Publicado
	CUI	Inclusão múltipla CUI Genérica Completa	Proposto
	FUI	Inclusão múltipla FUI XHTML/JavaScript	Publicado
	FUI	Inclusão múltipla FUI Java/JSP	Publicado
	FUI	Inclusão múltipla FUI VB/ASP	Publicado
	FUI	Inclusão múltipla FUI PHP	Publicado
	CIP	Editar Web	Publicado
	SUI	Editar Web SUI	Publicado
	CUI	Editar Web CUI Genérica Completa	Proposto
	FUI	Editar Web FUI XHTML/JavaScript	Publicado
	FUI	Editar Web FUI Java/JSP	Publicado
	FUI	Editar Web FUI VB/ASP	Publicado
	FUI	Editar Web FUI PHP	Publicado

Coleção / Categoria	Tipo	Asset	Situação
	CIP	Associação simples com lista reduzida Web	Publicado
	SUI	Associação simples com lista reduzida Web SUI	Publicado
	CUI	Associação simples com lista reduzida Web CUI Genérica Completa	Proposto
	FUI	Associação simples com lista reduzida Web FUI XHTML/JavaScript	Publicado
	FUI	Associação simples com lista reduzida Web FUI Java/JSP	Publicado
	FUI	Associação simples com lista reduzida Web FUI VB/ASP	Publicado
	FUI	Associação simples com lista reduzida Web FUI PHP	Publicado
	CIP	Associação simples com lista extensa Web	Publicado
	SUI	Associação simples com lista extensa Web SUI	Publicado
	CUI	Associação simples com lista extensa Web CUI Genérica Completa	Proposto
	FUI	Associação simples com lista extensa Web FUI XHTML/JavaScript	Publicado
	FUI	Associação simples com lista extensa Web FUI Java/JSP	Publicado
	FUI	Associação simples com lista extensa Web FUI VB/ASP	Publicado
	FUI	Associação simples com lista extensa Web FUI PHP	Publicado
	CIP	Associação dupla Web	Publicado
	SUI	Associação dupla Web SUI	Publicado
	CUI	Associação dupla Web CUI Genérica Completa	Proposto
	FUI	Associação dupla Web FUI XHTML/JavaScript	Publicado
	FUI	Associação dupla Web FUI Java/JSP	Publicado
	FUI	Associação dupla Web FUI VB/ASP	Publicado
	FUI	Associação dupla Web FUI PHP	Publicado
	CIP	Associação dupla com ordenação manual Web	Publicado
	SUI	Associação dupla com ordenação manual Web SUI	Publicado
	CUI	Associação dupla com ordenação manual Web CUI Genérica Completa	Proposto
	FUI	Associação dupla com ordenação manual Web FUI XHTML/JavaScript	Publicado
	FUI	Associação dupla com ordenação manual Web FUI Java/JSP	Publicado
	FUI	Associação dupla com ordenação manual Web FUI VB/ASP	Publicado
	FUI	Associação dupla com ordenação manual Web FUI PHP	Publicado
Diálogos padrões Web			
	CIP	Mensagem informativa Web	Publicado
	SUI	Mensagem informativa Web SUI	Publicado
	CUI	Mensagem informativa Web CUI Genérica Completa	Proposto
	FUI	Mensagem informativa Web FUI XHTML/JavaScript	Publicado
	CIP	Mensagem de erro Web	Publicado
	SUI	Mensagem de erro Web SUI	Publicado
	CUI	Mensagem de erro Web CUI Genérica Completa	Proposto
	FUI	Mensagem de erro Web FUI XHTML/JavaScript	Publicado
	CIP	Pergunta com respostas simples Web	Publicado

Coleção / Categoria	Tipo	Asset	Situação
	SUI	Pergunta com respostas simples Web SUI	Publicado
	CUI	Pergunta com respostas simples Web CUI Genérica Completa	Proposto
	FUI	Pergunta com respostas simples Web FUI XHTML/JavaScript	Publicado
	CIP	Pergunta com respostas complexas Web	Publicado
	SUI	Pergunta com respostas complexas Web SUI	Publicado
	CUI	Pergunta com respostas complexas Web CUI Genérica Completa	Proposto
	FUI	Pergunta com respostas complexas Web FUI XHTML/JavaScript	Publicado
Van Duyne [Duyne et al. 2002]			
Pattern Group C - Creating a Powerful Homepage			
	CIP	C1 Homepage Portal	Publicado
	SUI	C1 Homepage Portal SUI	Proposto
	CUI	C1 Homepage Portal CUI Completa	Proposto
	FUI	C1 Homepage Portal FUI PHP	Proposto
Pattern Group D - Writing and Managing Contents			
	CIP	D3 Headlines and Blurbs	Publicado
	SUI	D3 Headlines and Blurbs SUI	Proposto
	CUI	D3 Headlines and Blurbs CUI Completa	Proposto
	FUI	D3 Headlines and Blurbs FUI PHP	Proposto
Pattern Group H - Helping Customers Complete Tasks			
	CIP	H2 Sign-In/New Account	Publicado
	SUI	H2 Sign-In/New Account SUI	Proposto
	CUI	H2 Sign-In/New Account CUI Completa	Proposto
	FUI	H2 Sign-In/New Account FUI PHP	Proposto
	CIP	H7 Frequently Asked Questions	Publicado
	SUI	H7 Frequently Asked Questions SUI	Proposto
	CUI	H7 Frequently Asked Questions CUI Completa	Proposto
	FUI	H7 Frequently Asked Questions FUI PHP	Proposto
Pattern Group J - Making Site Search Fast and Relevant			
	CIP	J1 Search Action Module	Publicado
	SUI	J1 Search Action Module SUI	Proposto
	CUI	J1 Search Action Module CUI Completa	Proposto
	FUI	J1 Search Action Module FUI PHP	Proposto
	CIP	J2 Straightforward Search Forms	Publicado
	SUI	J2 Straightforward Search Forms SUI	Proposto
	CUI	J2 Straightforward Search Forms CUI Completa	Proposto
	FUI	J2 Straightforward Search Forms FUI PHP	Proposto
	CIP	J3 Organized Search Results	Publicado
	SUI	J3 Organized Search Results SUI	Proposto

Coleção / Categoria	Tipo	Asset	Situação
	CUI	J3 Organized Search Results CUI Completa	Proposto
	FUI	J3 Organized Search Results FUI PHP	Proposto
Pattern Group K - Making Navigation Easy			
	CIP	K2 Navigation Bar	Publicado
	SUI	K2 Navigation Bar SUI	Proposto
	CUI	K2 Navigation Bar CUI Completa	Proposto
	FUI	K2 Navigation Bar FUI PHP	Proposto
	CIP	K3 Tab Rows	Publicado
	SUI	K3 Tab Rows SUI	Proposto
	CUI	K3 Tab Rows CUI Completa	Proposto
	FUI	K3 Tab Rows FUI XHTML/JavaScript	Proposto
	CIP	K4 Action Buttons	Publicado
	SUI	K4 Action Buttons SUI	Proposto
	CUI	K4 Action Buttons CUI Completa	Proposto
	FUI	K4 Action Buttons FUI XHTML/JavaScript	Proposto
	CIP	K6 Location Bread Crumbs	Publicado
	SUI	K6 Location Bread Crumbs SUI	Proposto
	CUI	K6 Location Bread Crumbs CUI Completa	Proposto
	FUI	K6 Location Bread Crumbs FUI XHTML/JavaScript	Proposto
	CIP	K12 Preventing Errors	Publicado
	SUI	K12 Preventing Errors SUI	Proposto
	CUI	K12 Preventing Errors CUI Completa	Proposto
	FUI	K12 Preventing Errors FUI XHTML/JavaScript	Proposto
	CIP	K13 Meaningful Error Messages	Publicado
	SUI	K13 Meaningful Error Messages SUI	Proposto
	CUI	K13 Meaningful Error Messages CUI Completa	Proposto
	FUI	K13 Meaningful Error Messages FUI XHTML/JavaScript	Proposto
Van Welie [Welie 2005]			
Ecommerce			
	CIP	Login	Publicado
	SUI	Login SUI	Proposto
	CUI	Login CUI Completa	Proposto
	FUI	Login FUI XHTML/JavaScript	Proposto
Navigation			
	CIP	Breadcrumbs	Publicado
	SUI	Breadcrumbs SUI	Proposto
	CUI	Breadcrumbs CUI Completa	Proposto
	FUI	Breadcrumbs FUI XHTML/JavaScript	Proposto

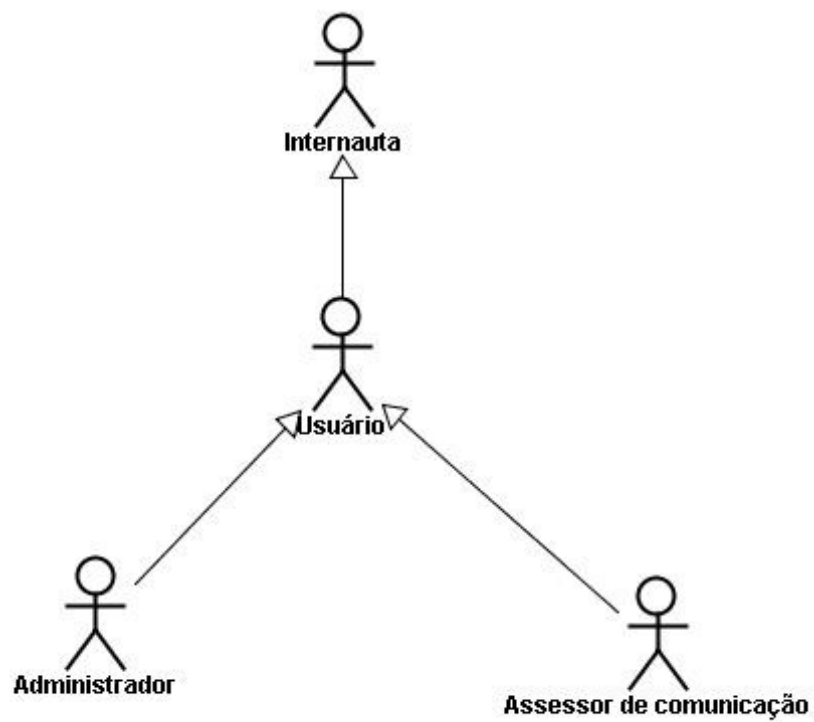
Coleção / Categoria	Tipo	Asset	Situação
	CIP	Double Tab Navigation	Publicado
	SUI	Double Tab Navigation SUI	Proposto
	CUI	Double Tab Navigation CUI Completa	Proposto
	FUI	Double Tab Navigation FUI XHTML/JavaScript	Proposto
	CIP	Fly-out Menu	Publicado
	SUI	Fly-out Menu SUI	Proposto
	CUI	Fly-out Menu CUI Completa	Proposto
	FUI	Fly-out Menu FUI XHTML/JavaScript	Proposto
	CIP	Main Navigation	Publicado
	SUI	Main Navigation SUI	Proposto
Searching			
	CIP	Site Map	Publicado
	SUI	Site Map SUI	Proposto
	CIP	Search Results	Publicado
	SUI	Search Results SUI	Proposto
	CUI	Search Results CUI Completa	Proposto
	FUI	Search Results FUI XHTML/JavaScript	Proposto
	CIP	FAQ	Publicado
	SUI	FAQ SUI	Proposto
	CIP	Search Box	Publicado
	SUI	Search Box SUI	Proposto
	CUI	Search Box CUI Completa	Proposto
	FUI	Search Box FUI XHTML/JavaScript	Proposto
Basic Page Types			
	CIP	Input Error Message	Publicado
	SUI	Input Error Message SUI	Proposto
	CUI	Input Error Message CUI Completa	Proposto
	FUI	Input Error Message FUI XHTML/JavaScript	Proposto
	CIP	Homepage	Publicado
	SUI	Homepage SUI	Proposto
	CIP	Form	Publicado
	SUI	Form SUI	Proposto
	CUI	Form CUI Completa	Proposto
	FUI	Form FUI XHTML/JavaScript	Proposto
Managing Collections			
	CIP	Table Sorter	Publicado
	SUI	Table Sorter SUI	Proposto
	CUI	Table Sorter CUI Completa	Proposto

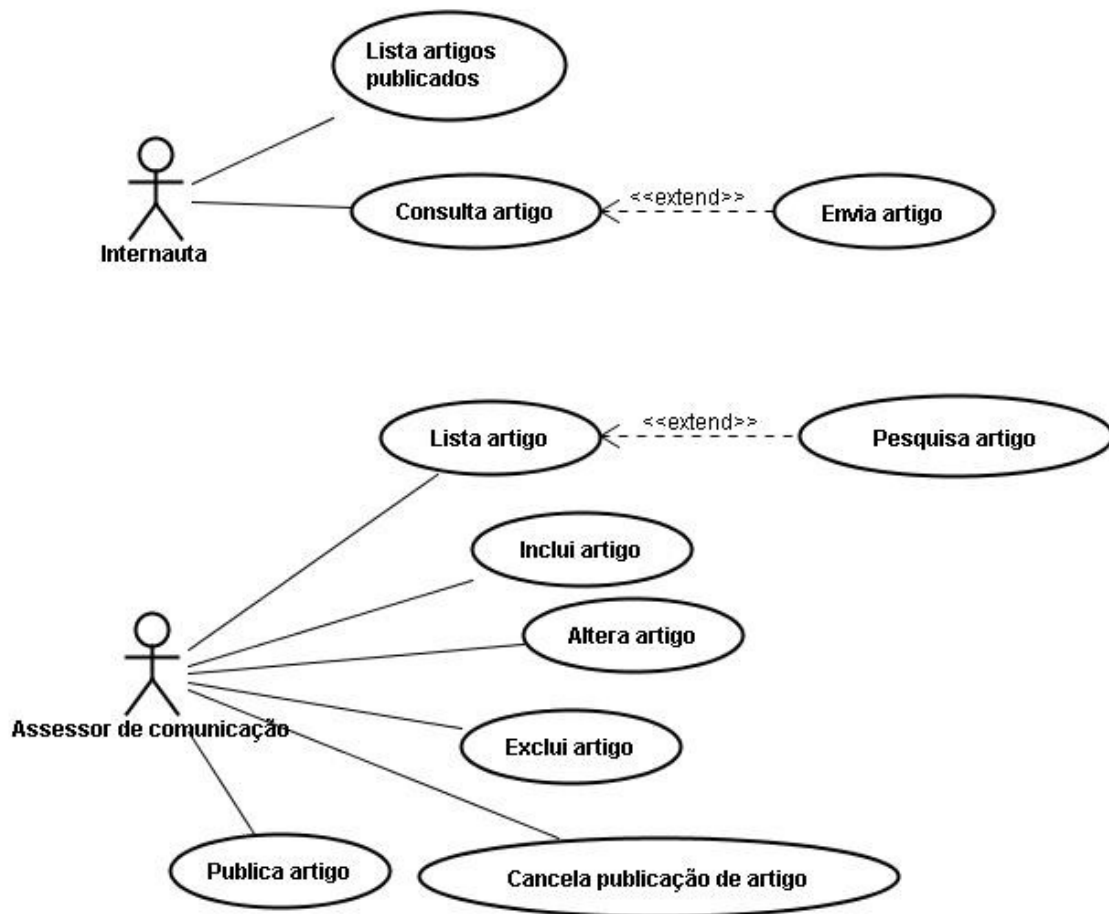
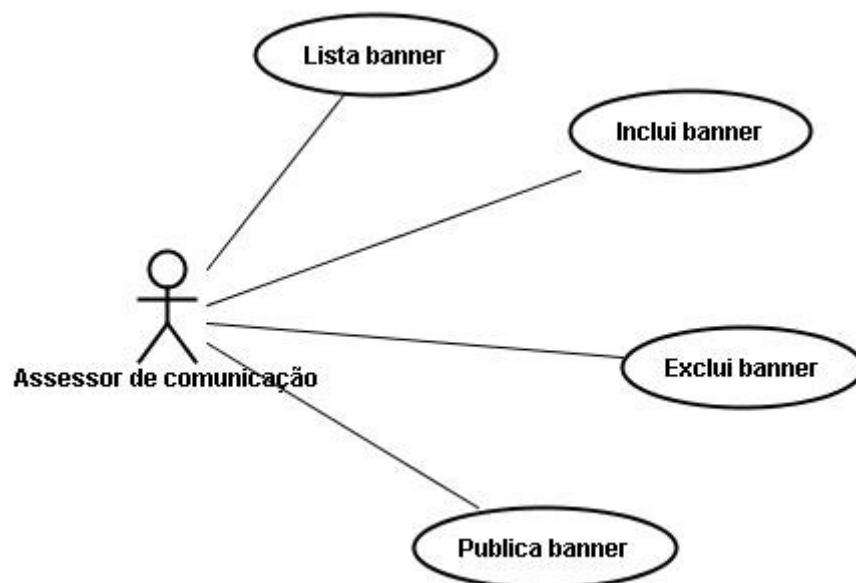
Coleção / Categoria	Tipo	Asset	Situação
	FUI	Table Sorter FUI XHTML/JavaScript	Proposto
	CIP	View	Publicado
	SUI	View SUI	Proposto
	CUI	View CUI Completa	Proposto
	FUI	View FUI XHTML/JavaScript	Proposto
	CIP	Parts Selector	Publicado
	SUI	Parts Selector SUI	Proposto
	CUI	Parts Selector CUI Completa	Proposto
	FUI	Parts Selector FUI XHTML/JavaScript	Proposto
	CIP	List Sorter	Publicado
	SUI	List Sorter SUI	Proposto
	CUI	List Sorter CUI Completa	Proposto
	FUI	List Sorter FUI XHTML/JavaScript	Proposto
Page Elements			
	CIP	Date Selector	Publicado
	SUI	Date Selector SUI	Proposto
	CUI	Date Selector CUI Completa	Proposto
	FUI	Date Selector FUI XHTML/JavaScript	Proposto
	CIP	News Box	Publicado
	SUI	News Box SUI	Proposto
	CIP	Home Link	Publicado
	SUI	Home Link SUI	Proposto
	CUI	Home Link CUI Completa	Proposto
	FUI	Home Link FUI XHTML/JavaScript	Proposto
	CIP	To-the-top Link	Publicado
	SUI	To-the-top Link SUI	Proposto
	CUI	To-the-top Link CUI Completa	Proposto
	FUI	To-the-top Link FUI XHTML/JavaScript	Proposto
	CIP	Constraint Input	Publicado
	SUI	Constraint Input SUI	Proposto
	CIP	Poll	Publicado
	SUI	Poll SUI	Proposto
Basic Interactions			
	CIP	Wizard	Publicado
	SUI	Wizard SUI	Proposto
	CUI	Wizard CUI Completa	Proposto
	FUI	Wizard FUI XHTML/JavaScript	Proposto
	CIP	Tabs	Publicado

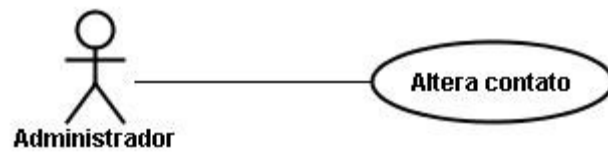
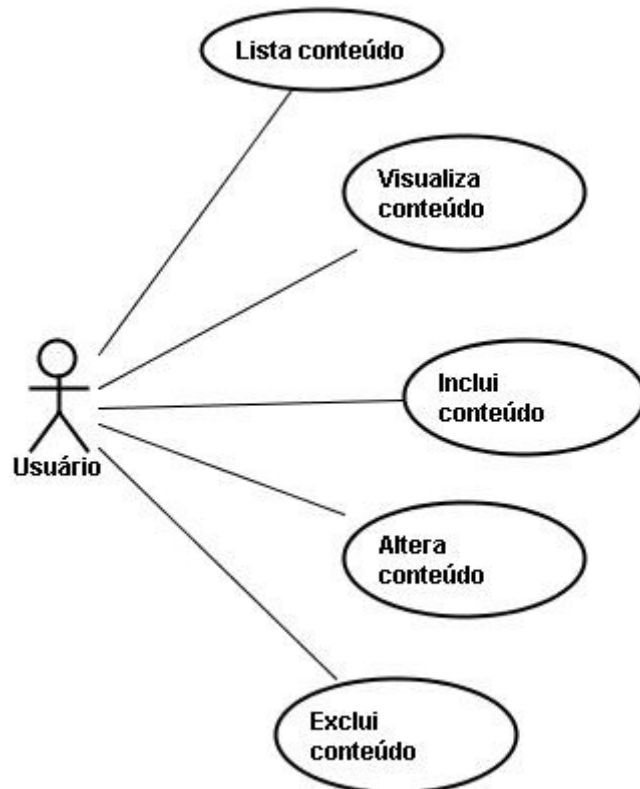
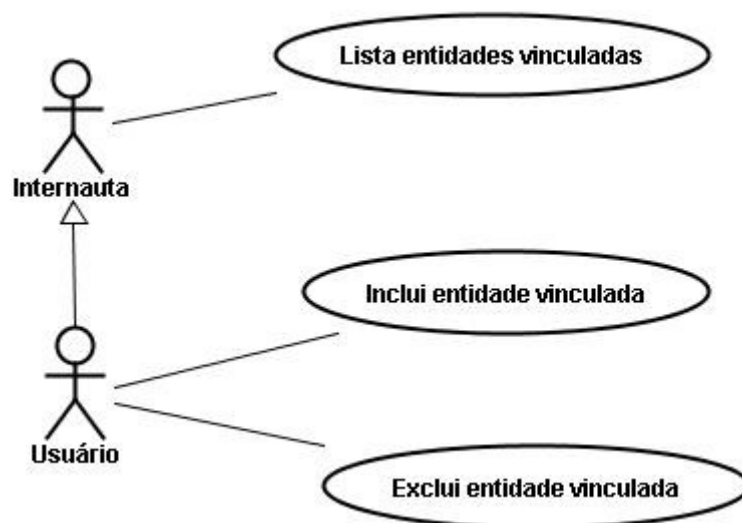
Coleção / Categoria	Tipo	Asset	Situação
	SUI	Tabs SUI	Proposto
	CUI	Tabs CUI Completa	Proposto
	FUI	Tabs FUI XHTML/JavaScript	Proposto
	CIP	Action Button	Publicado
	SUI	Action Button SUI	Proposto
	CUI	Action Button CUI Completa	Proposto
	FUI	Action Button FUI XHTML/JavaScript	Proposto
	CIP	Paging	Publicado
	SUI	Paging SUI	Proposto
	CUI	Paging CUI Completa	Proposto
	FUI	Paging FUI XHTML/JavaScript	Proposto
Visual Design			
	CIP	Liquid Layout	Publicado
	SUI	Liquid Layout SUI	Proposto
	CUI	Liquid Layout CUI Completa	Proposto
	FUI	Liquid Layout FUI XHTML/JavaScript	Proposto
	CIP	Alternating Row Colors	Publicado
	SUI	Alternating Row Colors SUI	Proposto
	CUI	Alternating Row Colors CUI Completa	Proposto
	FUI	Alternating Row Colors FUI XHTML/JavaScript	Proposto

APÊNDICE E – CASOS DE USO DO ESTUDO DE CASO

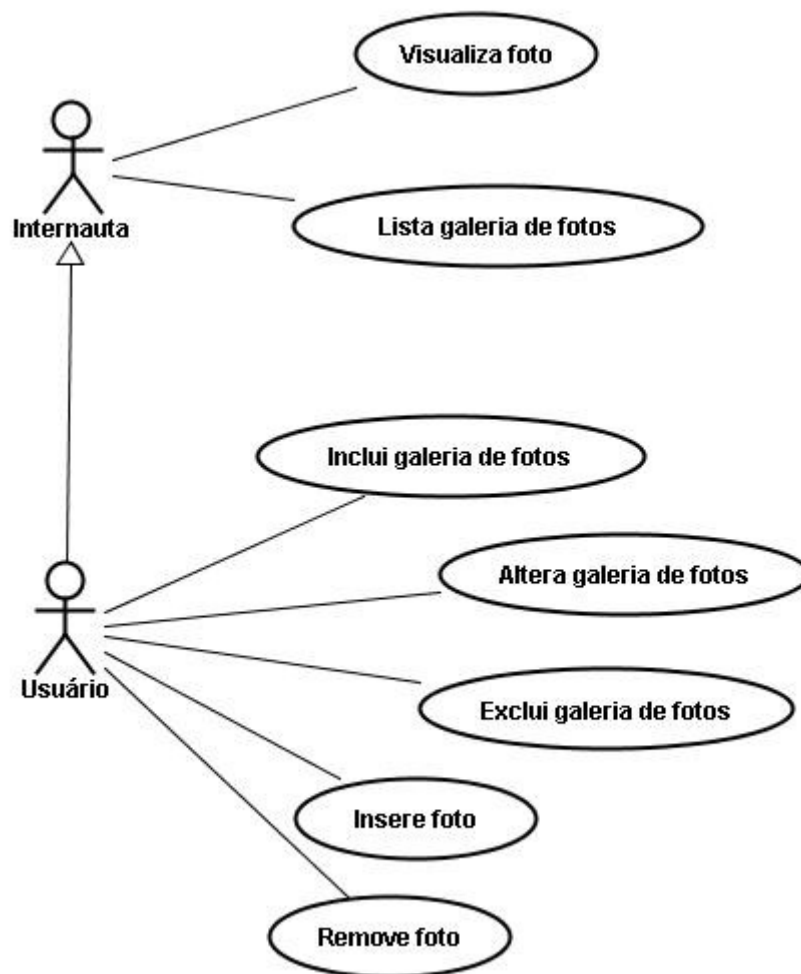
Atores do sistema



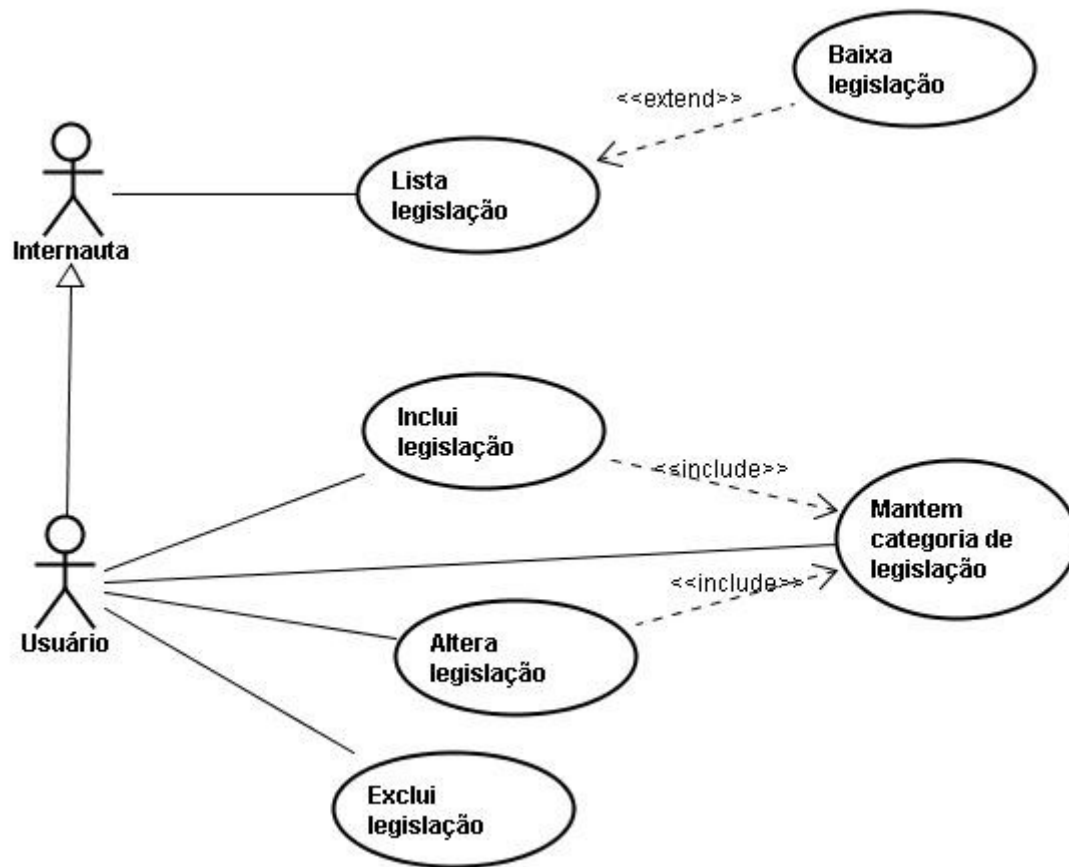
Pacote: artigo**Pacote: banner**

Pacote: contato**Pacote: conteudo****Pacote: entidade**

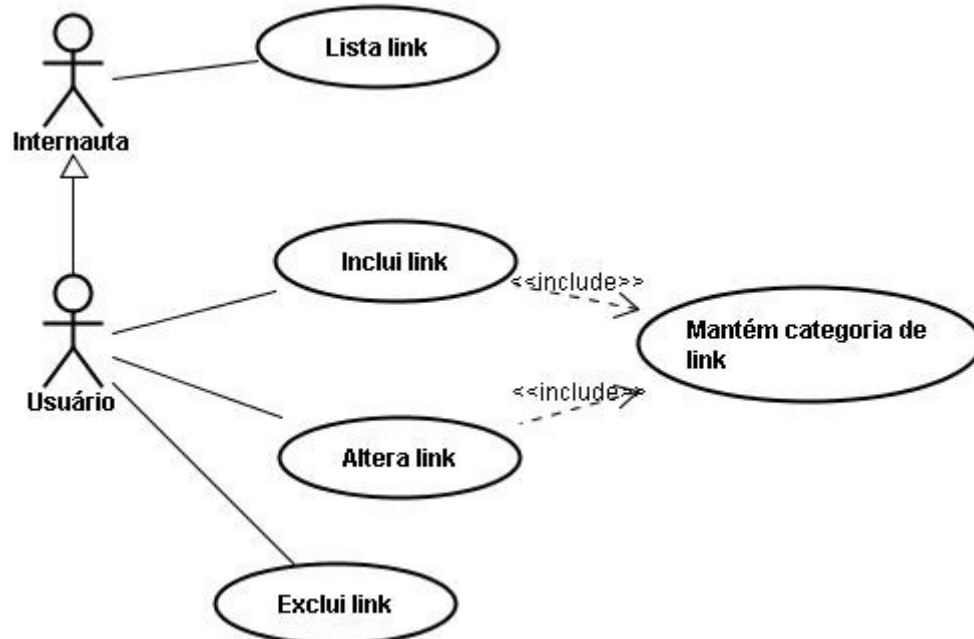
Pacote: foto



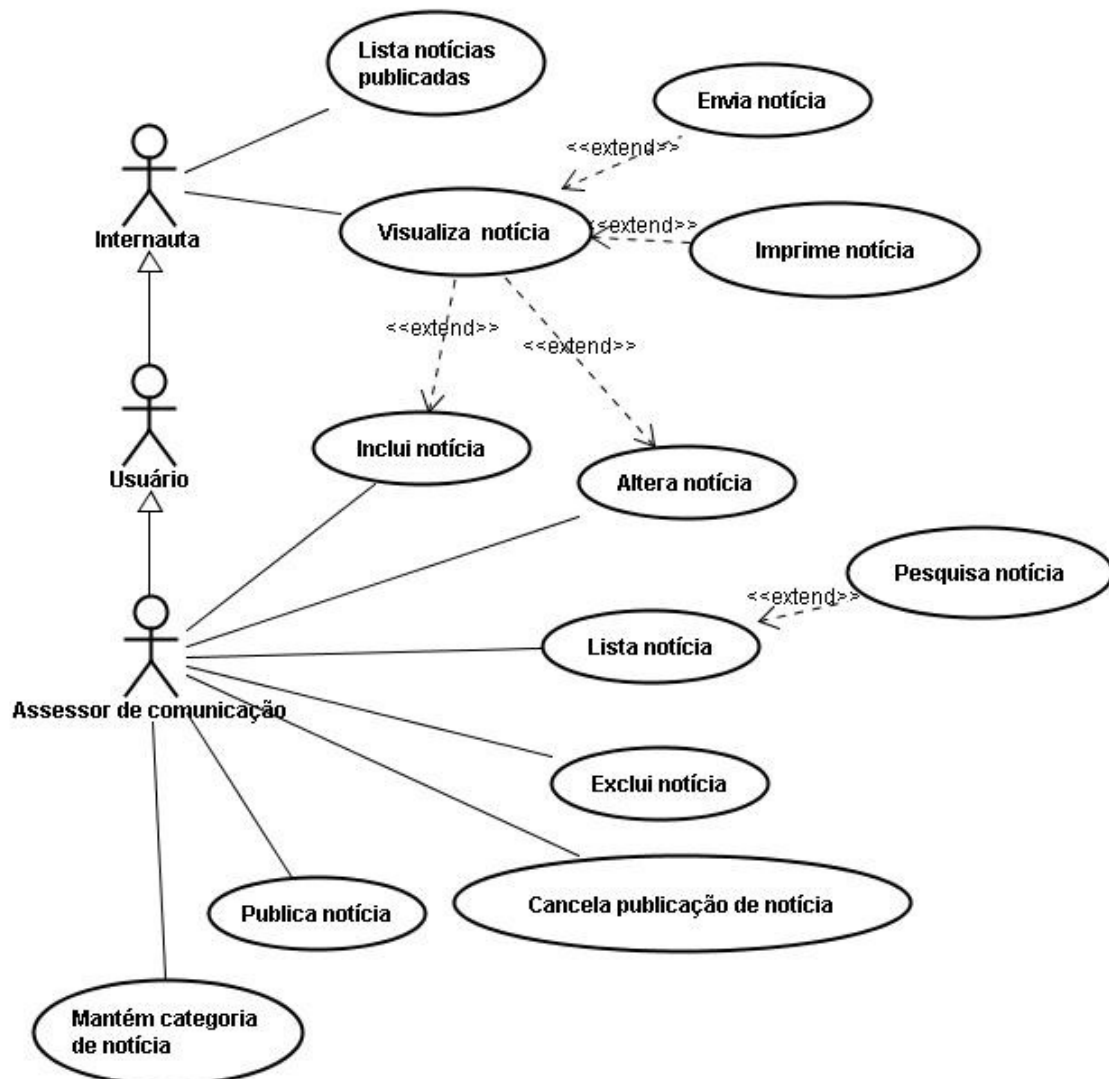
Pacote: legislacao



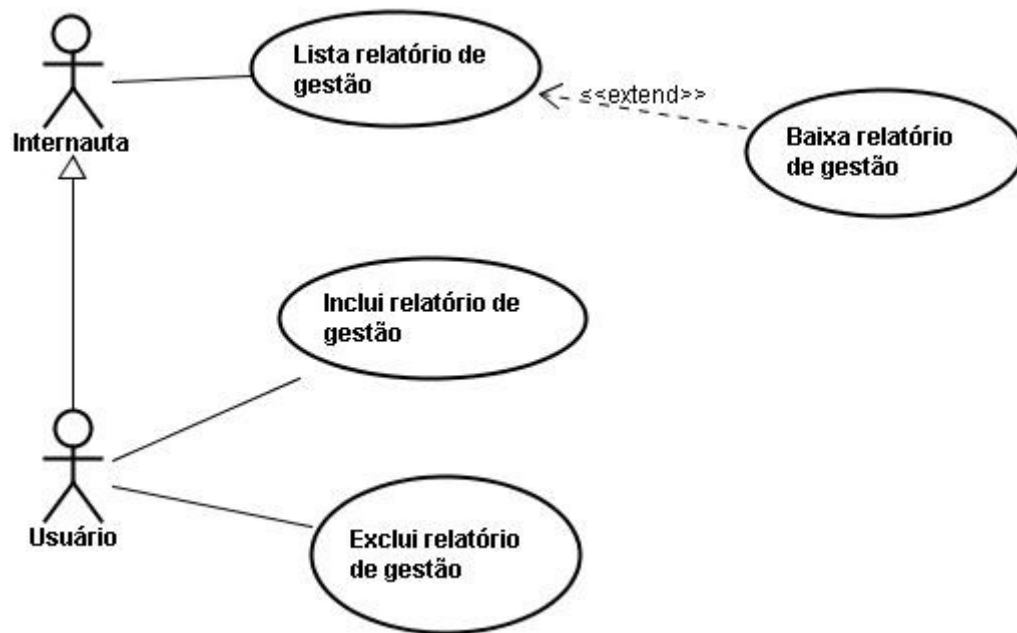
Pacote: link



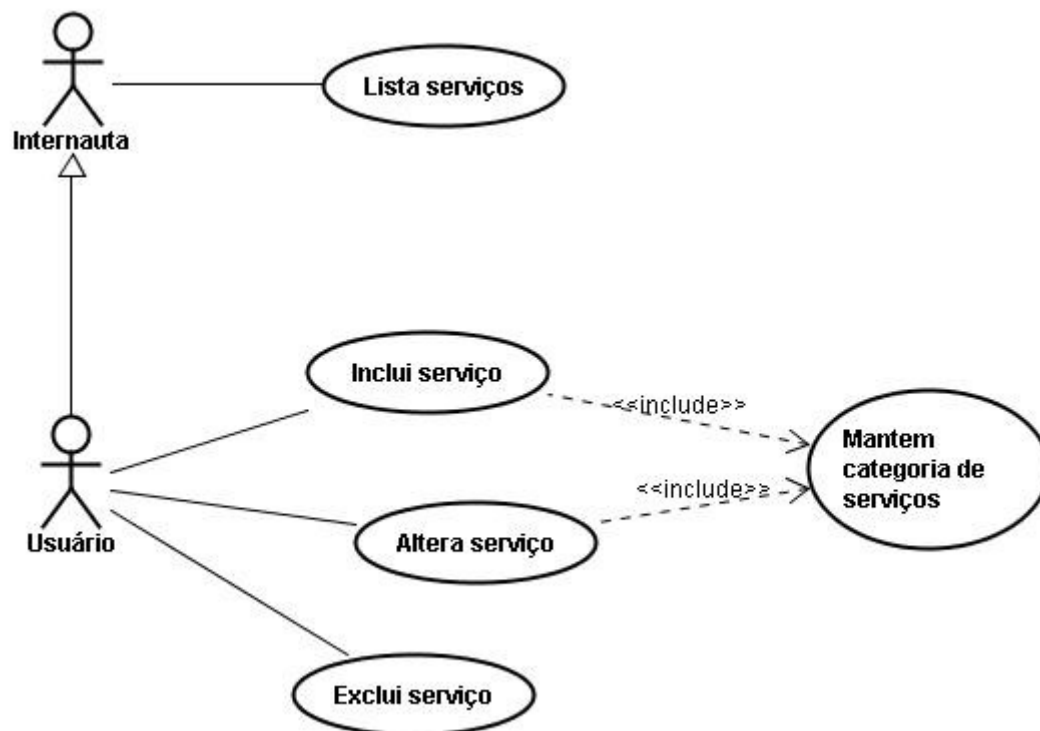
Pacote: notícia



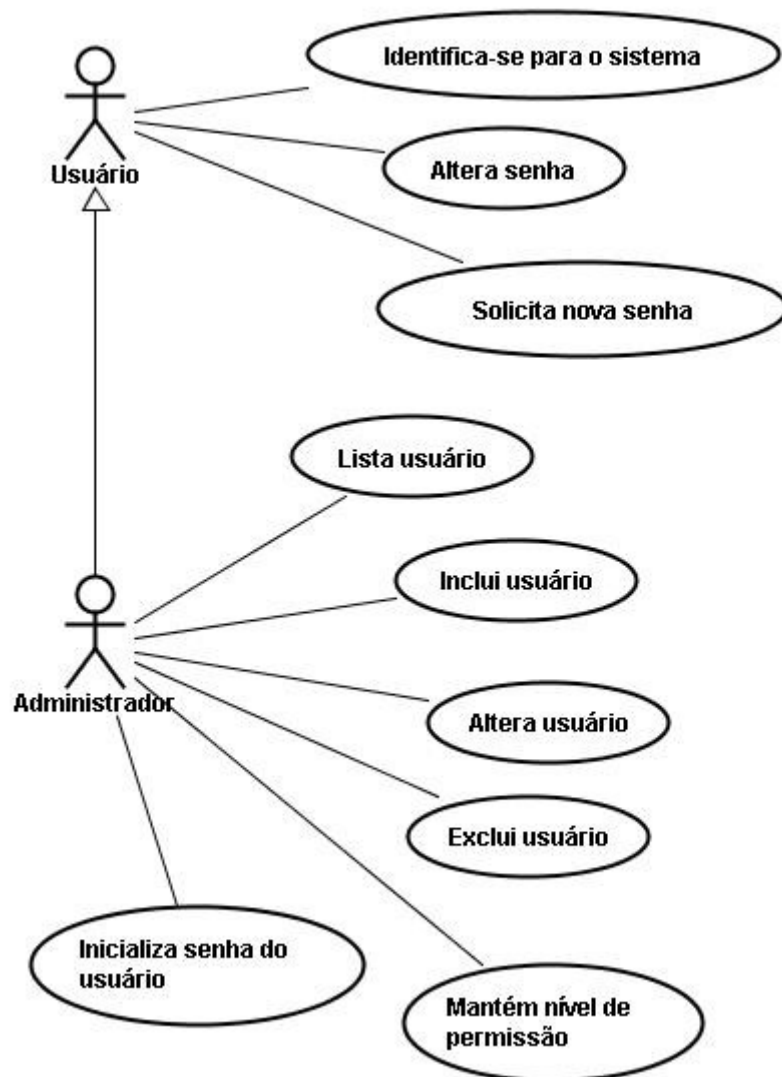
Pacote: relatório



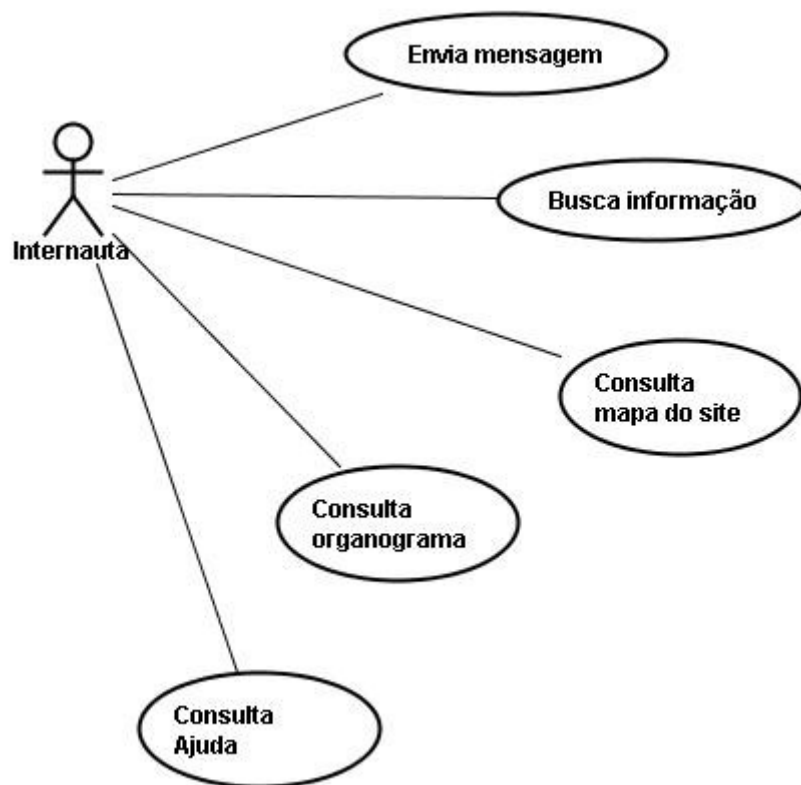
Pacote: serviço



Pacote: acesso



Pacote: outros



APÊNDICE F – EUCP DOS CASOS DE USO DO ESTUDO DE CASO

A tabela abaixo apresenta a tabulação dos pontos de caso de uso do sistema utilizado no estudo de caso.

A colunas abaixo de fator de reuso representam as fases do processo de desenvolvimento e possuem a seguinte notação: A-Análise, P-Projeto, C-Construção, T-Teste e G-Gerenciamento.

Tabela F.1: EUCP dos casos de uso do estudo de caso.

Nro	Pacote	Objetivo ao nível de tarefa	Nível de Reuso	UCP	Fator de Reuso (RF)					EUCP
					A	P	C	T	G	
4	artigo	Consulta artigo	9 - FUI	5	0,8	0,8	0,4	0,0	0,0	2,80
5	artigo	Envia artigo	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
9	artigo	Lista artigos publicados	9 - FUI	5	0,8	0,8	0,2	0,0	0,0	3,20
8	artigo	Lista artigo	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
10	artigo	Pesquisa artigo	9 - FUI	5	0,8	0,6	0,4	0,2	0,0	2,85
7	artigo	Inclui artigo	9 - FUI	5	0,8	0,6	0,4	0,0	0,0	2,95
2	artigo	Altera artigo	9 - FUI	5	0,8	0,6	0,4	0,0	0,0	2,95
6	artigo	Exclui artigo	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
11	artigo	Publica artigo	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
3	artigo	Cancela publicação de artigo	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
12	banner	Lista banner	9 - FUI	5	0,8	0,8	0,4	0,4	0,0	2,60
13	banner	Inclui banner	9 - FUI	5	0,8	0,6	0,4	0,4	0,0	2,75
14	banner	Exclui banner	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
15	banner	Publica banner	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
16	contato	Altera contato	9 - FUI	5	0,8	0,8	0,8	0,8	0,0	1,60
17	conteudo	Lista conteúdo	9 - FUI	5	0,8	0,8	0,4	0,4	0,0	2,60
18	conteudo	Visualiza conteúdo	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
19	conteudo	Inclui conteúdo	9 - FUI	5	0,8	0,6	0,4	0,4	0,0	2,75
20	conteudo	Altera conteúdo	9 - FUI	5	0,8	0,6	0,4	0,4	0,0	2,75
21	conteudo	Exclui conteúdo	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
22	entidade	Lista entidades vinculadas	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
23	entidade	Inclui entidade vinculada	9 - FUI	5	0,8	0,6	0,6	0,4	0,0	2,35
24	entidade	Exclui entidade vinculada	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
26	foto	Lista galeria de fotos	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
25	foto	Visualiza foto	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
27	foto	Inclui galeria de fotos	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
28	foto	Altera galeria de fotos	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
29	foto	Exclui galeria de fotos	9 - FUI	5	0,8	0,8	0,8	0,6	0,0	1,70
30	foto	Insere foto	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00

Nro	Pacote	Objetivo ao nível de tarefa	Nível de Reuso	UCP	Fator de Reuso (RF)					EUCP
					A	P	C	T	G	
4	artigo	Consulta artigo	9 - FUI	5	0,8	0,8	0,4	0,0	0,0	2,80
31	foto	Remove foto	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
73	geral	Busca informação	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
72	geral	Envia mensagem	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
76	geral	Consulta Ajuda	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
74	geral	Consulta mapa do site	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
75	geral	Consulta organograma	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
32	legislacao	Lista legislação	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
33	legislacao	Baixa legislação	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
34	legislacao	Inclui legislação	9 - FUI	5	0,8	0,6	0,6	0,4	0,0	2,35
35	legislacao	Altera legislação	9 - FUI	5	0,8	0,6	0,6	0,4	0,0	2,35
36	legislacao	Exclui legislação	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
37	legislacao	Mantem categoria de legislação	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
38	link	Lista link	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
39	link	Inclui link	9 - FUI	5	0,8	0,6	0,6	0,4	0,0	2,35
40	link	Altera link	9 - FUI	5	0,8	0,6	0,6	0,4	0,0	2,35
41	link	Exclui link	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
42	link	Mantém categoria de link	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
43	noticia	Lista notícias publicadas	9 - FUI	5	0,8	0,6	0,6	0,4	0,0	2,35
44	noticia	Visualiza notícia	9 - FUI	5	0,8	0,6	0,6	0,4	0,0	2,35
45	noticia	Envia notícia	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
46	noticia	Imprime notícia	9 - FUI	5	0,8	0,4	0,4	0,2	0,0	3,00
47	noticia	Lista notícia	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
48	noticia	Pesquisa notícia	9 - FUI	5	0,8	0,4	0,4	0,2	0,0	3,00
49	noticia	Inclui notícia	9 - FUI	5	0,6	0,4	0,2	0,0	0,0	3,70
50	noticia	Altera notícia	9 - FUI	5	0,6	0,4	0,2	0,0	0,0	3,70
51	noticia	Exclui notícia	9 - FUI	5	0,8	0,6	0,6	0,2	0,0	2,45
52	noticia	Publica notícia	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
53	noticia	Cancela publicação de notícia	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
1	noticia	Mantém categoria de notícia	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
54	relatorio	Lista relatório de gestão	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
55	relatorio	Baixa relatório de gestão	0 - Nenhum	5	0,0	0,0	0,0	0,0	0,0	5,00
56	relatorio	Inclui relatório de gestão	9 - FUI	5	0,8	0,6	0,6	0,4	0,0	2,35
57	relatorio	Exclui relatório de gestão	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
58	servico	Lista serviços	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
59	servico	Inclui serviço	9 - FUI	5	0,8	0,6	0,6	0,4	0,0	2,35
60	servico	Altera serviço	9 - FUI	5	0,8	0,6	0,6	0,4	0,0	2,35
61	servico	Exclui serviço	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
62	servico	Mantem categoria de serviços	9 - FUI	5	0,8	0,8	0,8	0,4	0,0	1,80
63	usuario	Identifica-se para o sistema	9 - FUI	5	0,8	0,8	0,6	0,6	0,0	2,10
64	usuario	Altera senha	9 - FUI	5	0,8	0,8	0,6	0,6	0,0	2,10
65	usuario	Solicita nova senha	9 - FUI	5	0,8	0,8	0,6	0,6	0,0	2,10
70	usuario	Inicializa senha do usuário	10 - XUI	5	0,8	0,8	0,8	0,8	0,0	1,60
66	usuario	Lista usuário	10 - XUI	5	0,8	0,8	0,8	0,8	0,0	1,60
67	usuario	Inclui usuário	10 - XUI	5	0,8	0,8	0,8	0,8	0,0	1,60
68	usuario	Altera usuário	10 - XUI	5	0,8	0,8	0,8	0,8	0,0	1,60
69	usuario	Exclui usuário	10 - XUI	5	0,8	0,8	0,8	0,8	0,0	1,60
71	usuario	Mantém nível de permissão	10 - XUI	5	0,8	0,8	0,8	0,8	0,0	1,60
Total				380						213,2

2. O sistema apresenta as notícias selecionadas e aguarda que o assessor de comunicação complete as informações para publicação

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX

E-MAIL SENHA AJUDA SAÍDA

Módulo Administração

Publicar notícias

Publicar

Cancelar

Ajuda

Notícia a ser publicada	Data e Hora publicação	Destaque	
Título da notícia 1	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 2	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 3	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 4	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 5	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 6	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 7	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 8	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 9	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 10	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 11	20/12/2006 Calend 08:00	Nenhuma	Remover
Título da notícia 12	20/12/2006 Calend 08:00	Nenhuma	Remover

Noticias

Objeto 2.

Objeto 2.

Objeto 2.

Objeto 2.

Nome do operador:
Fulano de Tal e tal

3. O assessor de comunicação fornece as informações para publicação

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX E-MAIL SENHA AJUDA SAÍDA

Módulo Administração ▼

Publishar notícias

Publicar Cancelar Ajuda

Notícia a ser publicada	Data e Hora publicação	Destaque	
Título da notícia 1	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 2	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 3	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 4	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 5	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 6	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 7	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 8	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 9	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 10	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 11	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 12	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover

Notícias

Objeto 2.

Objeto 2.

Objeto 2.

Objeto 2.

Nome do operador:
Fulano de Tal e tal

4. Eventualmente, o assessor de comunicação remove uma notícia do conjunto de notícias a serem publicadas

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX E-MAIL SENHA AJUDA SAÍDA

Módulo Administração ▼

Publicar notícias

Publicar Cancelar Ajuda

Notícia a ser publicada	Data e Hora publicação	Destaque	
Título da notícia 1	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 2	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 3	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 4	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 5	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 6	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 7	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 8	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 9	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 10	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 11	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover
Título da notícia 12	20/12/2006 Calend 08:00 ▼	Nenhuma ▼	Remover

Notícias

Objeto 2.

Objeto 2.

Objeto 2.

Objeto 2.






Nome do operador:
Fulano de Tal e tal

5. O assessor de comunicação submete as notícias para publicação

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX E-MAIL SENHA AJUDA SAÍDA

Módulo Administração  **Publicar notícias**

 **Publicar** **Cancelar** **Ajuda**

Notícia a ser publicada	Data e Hora publicação	Destaque	
<u>Título da notícia 1</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 2</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 3</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 4</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 5</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 6</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 7</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 8</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 9</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 10</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 11</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover
<u>Título da notícia 12</u>	20/12/2006 Calend  08:00 	Nenhuma 	Remover

Notícias

Objeto 2.

Objeto 2.

Objeto 2.

Objeto 2.

Nome do operador:
Fulano de Tal e tal

6. O sistema publica as notícias conforme definido

7. O caso de uso é encerrado

Fluxo alternativo

1a. Nenhuma notícia foi selecionada

1a1. Sistema solicita que seja selecionada pelo menos uma notícia e retorna ao passo 1

[illegible]

- 4a. Existe somente uma notícia para ser removida
- 4a1. Sistema solicita confirmação da remoção
- 4a2. O assessor de comunicação confirma a remoção
- 4a3. O caso de uso é encerrado
- 4a2a. O assessor de comunicação não confirma a remoção
- 4a2a1. O caso de uso retorna ao passo 3

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX E-MAIL SENHA AJUDA SAÍDA

Módulo Administração

Publicar notícias

Publicar **Cancelar** **Ajuda**

Notícia a ser publicada	Data e Hora publicação	Destaque
<u>Título da notícia 1</u>	20/12/2006 Calend 08:00	Nenhum
		Remover

Notícias:

- Objeto 2.
- Objeto 2.
- Objeto 2.
- Objeto 2.

Nome do operador:
Fulano de Tal e tal

Ao remover esta notícia, o processo de publicação de notícias será encerrado.
Confirma a remoção da notícia?

OK **Cancelar**

5a. Os dados de publicação informados não são válidos ou estão incompletos

5a1. Sistema comunica este fato ao assessor de comunicação e retorna ao passo

3

SISTEMA DE ATUALIZAÇÃO DINÂMICA DO SITE XXXXXXXXXXXXXXXX E-MAIL SENHA AJUDA SAÍDA

Módulo Administração ▼

Publishar notícias

Publicar **Cancelar** **Ajuda**

Notícia a ser publicada	Data e Hora publicação	Destaque	
<u>Título da notícia 1</u>	<input type="text"/> Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 2</u>	20/12/2006 Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 3</u>	20/12/2006 Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 4</u>	<input type="text"/> Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 5</u>	<input type="text"/> Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 6</u>	<input type="text"/> Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 7</u>	<input type="text"/> Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 8</u>	20/12/2006 Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 9</u>	20/12/2006 Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 10</u>	20/12/2006 Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 11</u>	20/12/2006 Calend 08:00 ▼	Nenhum ▼	Remover
<u>Título da notícia 12</u>	20/12/2006 Calend 08:00 ▼	Nenhum ▼	Remover

Nome do operador:
Fulano de Tal e tal

Informe a data de publicação
OK