UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JAMILE DE SOUZA MARTINS

ReUse
Uma Ferramenta de Suporte a Reuso

Projeto de Diplomação

Prof. Dr. Marcelo Soares Pimenta
Orientador

Porto Alegre, junho de 2008

“Só os chatos sobrevivem.”
— FÁBIO PETRILLO

AGRADECIMENTOS

Agradeço, primeiramente aos meus pais, Newton e Dione, meus maiores motivadores, tanto pelas palavras de carinho nos momentos difíceis, quanto pelo maravilhoso exemplo, que sempre me deram a vida inteira. Essa vitória é tão de vocês quanto minha.

Agradeço à minha família, que por tantas vezes se privou da minha companhia em nome dos trabalhos, estudo para provas e tantas outras atividades acadêmicas ao longo desses cinco anos.

Ao meu namorado e companheiro Rodrigo, que me ajudou a revisar esse texto, aprendendo junto comigo e colaborando de todas as formas possíveis para finalização desse trabalho. Obrigada pelo amor, pelas palavras de carinho, pela paciência, e principalmente por nunca me deixar esmorecer.

Agradeço também à todos os amigos e colegas que durante todo esse tempo viveram junto comigo a emoção de cada final de semestre, as noites em frente ao computador para terminar algum trabalho, os momentos de pavor na biblioteca antes das provas. Em especial, gostaria de agradecer às colegas Dâmara e Mariane, grandes amigas que fiz nesse período, que me ensinaram mais do que podem imaginar. Agradecimentos especiais aos colegas de barra Jean (Juzám), Eduardo, Leonardo, André, Soraya, Ricardo Nolde, Filipe, Pablo, Fernando, Pedro e tantos outros, sempre compartilhando comigo dúvidas (muitas dúvidas) e soluções.

Ao meu orientador, Marcelo Pimenta, por me acompanhar na execução deste trabalho, pelos conselhos e principalmente pela confiança depositada.

Em especial, a duas pessoas que sequer se conhecem mas que foram fundamentais em todo esse processo: à Emily, por sempre ter acreditado em mim e ao Petrillo, por todas as lições tecnológicas e humanas que me ensinou.

Muito obrigada à todos vocês que participaram desta etapa. Eu não teria conseguido nada sozinha.

Obrigada!

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

OMG	Object Management Group
IU	Interface de Usuário
RAS	Reusable Asset Specification
CRF	Cameleon Reference Framework
UML	Unified Modeling Language
PU	Processo Unificado
GUI	Graphical User Interface
IO	Interaction Object
CIO	Concrete Interaction Object
AIO	Abstract Interaction Object
RUCP	Padrão de Caso de Uso Reificado
IDE	Integrated Development Environment

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O reuso de software é uma técnica bastante antiga na Engenharia de Software, porém, pouco colocada em prática. Dentre as razões técnicas que levam a esse problema, estão a falta de apoio metodológico para a prática do reuso e a falta de ferramentas que dêem suporte a uma abordagem mais abrangente de reuso.

O objetivo deste trabalho é propor uma ferramenta de suporte a reuso para elementos de IU (Interface de Usuário), em diversos níveis da aplicação e que permite reuso de artefatos desde casos de uso (o mais alto nível) até implementações (o nível mais concreto), passando por vários níveis de detalhamento, todos reusáveis.

Palavras-chave: Engenharia de software, reuso de software, casos de uso, interface de usuário.

ReUse
A Software Reuse Support Tool

ABSTRACT

Software reuse is a classic technique in Software Engineering although it is not as much used as should be. Such problem is due to some technical reasons like, for example, a lack of methodological support for reuse practice and a lack of tools that support software reuse in different, more including levels of abstraction.

The main goal of this work is to propose a software reuse tool to UI (User Interface) elements, in several aplication levels, varying from the well-known use cases (the most abstract level) to an UI implementation description (the most concrete level), with several intermediate levels corresponding to artifacts of different stages of UI design and construction, and all of them are reusable.

Keywords: software engineering, software reuse, use cases, user interfaces.

1 INTRODUÇÃO

O reuso de software é uma técnica antiga, existente desde os primórdios da Engenharia de Software (MCI 68) e vista por muitos autores como uma das mais recomendáveis para solucionar os problemas do desenvolvimento de sistemas. Porém, apesar de ser uma técnica muito antiga, conhecida e estudada, é pouco colocada efetivamente em prática nos ambientes de desenvolvimento de sistemas atuais.

Muitas são as razões para que isto ocorra. Questões de cunho técnico, organizacional, psicológico, sociológico e econômico são alguns dos que se interpõem à adoção da prática do reuso (TRA 88). Além disso, para que o reuso se torne possível, é necessário que i) os artefatos que compõem o sistema sejam construídos para serem reusados e ii) os artefatos que compõem o sistema sejam organizados de modo que sua busca e adaptação seja mais rápida do que a construção de um novo artefato (KRU 92).

Dessa forma, mesmo que uma instituição promova a prática do reuso, incentivando e investindo na criação de artefatos reusáveis, é preciso que estes artefatos sejam organizados em repositórios específicos, de modo a criar um ambiente propício para a prática do reuso. Se uma das etapas do desenvolvimento do sistema for procurar por soluções similares e artefatos que possam ser reusados em um repositório comum, o reuso de software pode se tornar um (bom) hábito.

O reuso pode se dar em vários momentos do ciclo de desenvolvimento e envolvendo diferentes elementos de *software*. Neste trabalho, o foco é o reuso de artefatos relacionados à interface com o usuário (IU).

A definição e a construção IU sofrem o mesmo problema de outros domínios do *software*, necessitando de abordagens mais sistemáticas e específicas de reuso. De acordo com a proposta de Augusto Moreira (MOR 2007) uma das abordagens para reuso de artefatos de software, mais especificamente artefatos de IU, pode ser a dirigida por casos de uso. Nessa abordagem, as possibilidades de reuso são analisadas sob a perspectiva de um caso de uso ao longo de todo o seu ciclo de vida.

Na modelagem dos casos de uso, adota-se a UML (*Unified Modeling Language*) (OMG 2007). Por ser uma linguagem padronizada para a comunicação, a UML se mostra eficaz e uma boa alternativa para propiciar uma estratégia de reuso. Os relacionamentos de extensão, especialização e inclusão, presentes entre os casos de uso na UML, dão suporte para que os conceitos básicos de reuso sejam aplicados.

O objetivo deste trabalho é propôr uma ferramenta que ofereça suporte à metodologia de reuso de IU proposta originalmente no trabalho de Augusto Moreira (MOR 2007), de modo a possibilitar o reuso em diversos níveis da aplicação, agrupados em três níveis essenciais: caso de uso, artefatos de especificação e artefatos de implementação. Além da definição dessa ferramenta será apresentado um protótipo e realizada uma análise das possíveis vantagens obtidas com a utilização de tal ferramenta no desenvolvimento de

sistemas.

A organização básica dos capítulos deste trabalho é descrita a seguir.

No capítulo 2 é realizada uma breve discussão sobre a prática do reuso, os tipos de reuso existentes, problemas em se realizar a prática do reuso e benefícios na sua adoção.

No capítulo 3 é descrito o problema, a abordagem proposta por Augusto (MOR 2007) em seu trabalho e de que modo este trabalho a complementa. Também são citadas algumas soluções similares encontradas e porque elas não abordam os níveis pretendidos nesse trabalho.

No capítulo 4 é apresentado o protótipo da ferramenta, contendo a descrição de suas funcionalidades gerais. Além disso é apresentado um exemplo de aplicação da ferramenta, incluindo a catalogação no repositório de um caso de uso e de vários artefatos associados até exemplos de busca de um elemento neste repositório.

No capítulo 5 são apresentadas as principais conclusões e sugestões de trabalhos futuros.

2 REUSO: CONCEITOS

Neste capítulo são apresentados os conceitos básicos de reuso de software, as principais tipos de reuso existentes e os problemas e benefícios em se adotar a prática do reuso.

2.1 Reuso de Software

Segundo Krueger (KRU 92), reuso de software é o processo de desenvolver sistemas de software a partir de software já existente, ao invés de "começar do zero".

É bastante comum, durante o processo de desenvolvimento de um sistema, que diversos problemas similares sejam resolvidos mais de uma vez, por diferentes desenvolvedores. A idéia básica do reuso é que estes problemas sejam resolvidos apenas uma vez, de modo a se ganhar tempo e, conseqüentemente, diminuir custos no processo de desenvolvimento. Além disso, com o reuso, o software evolui de forma natural: erros tendem a ser menos freqüentes e a qualidade do sistema como um todo aumenta gradativamente.

O reuso de software deve ser bastante planejado, pois a construção de artefatos reusáveis geralmente demanda mais tempo do que a construção dos demais artefatos. Para a utilização de artefatos reusáveis, é necessário que eles sejam construídos, planejados e documentados para este fim (TRA 88), de modo que o processo de busca, identificação e reutilização de um artefato leve menos tempo do que o processo de construção de um novo artefato (KRU 92).

Além disso, deve-se ter em mente que o reuso pode ser realizado em diferentes níveis de abstração, tais como código-fonte, padrões, artefatos de desenvolvimento, entre outros (AMB 2004). As seções a seguir apresentam os principais tipos de reuso existentes.

2.1.1 Tipos de Reuso

Existem diversas formas de reutilização de software. A mais conhecida é o reaproveitamento de código, mas elementos mais abstratos também podem ser reutilizados. Nas subseções que seguem são apresentadas as maneiras mais conhecidas de se utilizar o reuso no desenvolvimento de sistemas (MOR 2007).

2.1.1.1 Reuso de Artefatos

Artefatos de software são artefatos criados no decorrer do desenvolvimento. Eles podem ser casos de uso, documentos padrão, modelos de domínio, procedimentos e diretrizes e outras aplicações.

Esses artefatos podem ser utilizados de várias formas e em diversos níveis, desde o mais simples, em que o desenvolvedor reaproveita a idéia conceitual do artefato até

níveis mais abrangentes, onde o desenvolvedor reaproveita um artefato inteiro, tal qual foi desenvolvido, e o reutiliza em um novo projeto. Se o artefato reusado é resultante do processo de análise, o reuso é denominado **reuso de análise**; se for do projeto, é denominado **reuso de projeto**. Os artefatos reutilizados nos reuso de análise podem ser especificações, lógicas completas ou subconjuntos delas (tais como diagramas, descrição de dados ou descrição lógica de procedimentos); no reuso de projeto, os artefatos reutilizados podem ser as decisões e estratégias tomadas durante a etapa de projeto do sistema (ZIR 95).

Esse tipo de reutilização promove consistência entre projetos desenvolvidos e ainda reduz a carga de gerenciamento a cada novo projeto.

2.1.1.2 Reuso de Código

O reuso de código, sem dúvida, é uma das formas de reuso mais utilizadas por desenvolvedores. Nessa forma de reuso, partes de código fonte são reaproveitadas. Esse reaproveitamento pode acontecer, na melhor maneira, através do compartilhamento de classes, funções e rotinas comuns. De maneira mais "artesanal", acontece quando se copia um trecho do código fonte de uma aplicação em outra. Geralmente, por falta de ferramentas que cataloguem e forneçam uma busca eficiente, essa última é geralmente a forma de reuso de código (e reuso em geral) mais praticada.

A principal vantagem dessa forma de reuso é que menos código precisa ser escrito, já que existe código já pronto, logo, as chances de trechos de código reutilizados apresentarem erros diminui com o decorrer do tempo.

2.1.1.3 Reuso de Modelos ou Templates

Modelos ou *Templates* referem-se basicamente à documentação - utilização de um conjunto comum de *layouts* para documentos, modelos e código fonte. Logo o reuso de modelos é, basicamente, uma forma de reuso de documentação.

Ela pode ser tornar difícil, porém, na medida que os próprios desenvolvedores modificam a estrutura dos modelos para seu próprio uso, sem compartilhar as alterações com o resto da equipe. Entretanto, se esse tipo de problema não acontece, o reuso de modelos aumenta a consistência e a qualidade dos artefatos de desenvolvimento.

2.1.1.4 Reuso de Componentes

Componentes são artefatos pré-construídos, totalmente encapsulados e disponíveis para uso imediato para o desenvolvimento de aplicações. A principal diferença entre reutilização de componentes e de código fonte é que no reuso de componentes, não se tem acesso ao código fonte. O exemplo mais comuns de componentes são os componentes Java Beans.

As principais vantagens em se reutilizar componentes são que eles oferecem maior facilidade para a reutilização (os componentes são auto-suficientes e basta o desenvolvedor "conectar"o componente ao seu sistema para obter as funcionalidades oferecidas pelo componente) e que a ampla utilização de plataformas comuns incita a criação e a venda de componentes por terceiros.

A desvantagens na reutilização de componentes é que eles geralmente são pequenos, e como encapsulam um único conceito, um sistema que utiliza muitos componentes também tem uma biblioteca de componentes muito grande.

2.1.1.5 Reuso de Padrões ou Patterns

A idéia original de Padrões ou *Patterns* advém da Arquitetura e foi proposta por Christopher Alexander et al. (ALE 77). Apesar disso, tornaram-se populares na Engenharia de Software graças ao livro "Padrões de Projeto - Soluções reutilizáveis de Software Orientado a Objetos"(GoF - Gang of Four) (GAM 2000).

Cada padrão descreve um problema que ocorre seguidamente no desenvolvimento e descreve uma solução para esse problema, de forma que um desenvolvedor possa utilizar essa solução sempre que encontrar um problema compatível com o problema descrito pelo padrão (ALE 77). A descrição de um padrão tem pelo menos quatro campos definidos: Nome (nome que descreve o padrão), Problema (descrição do problema, de quando o padrão pode ser aplicado), Solução (descreve os elementos necessários, seus relacionamentos e papéis para se resolver o problema), e Consequências (os resultados obtidos ao se aplicar o padrão).

Como exemplo, tem-se o padrão *Singleton*, descrito no livro "Padrões de Projeto - Soluções reutilizáveis de Software Orientado a Objetos", citado acima. O nome do padrão, já se conhece: *Singleton*. O problema a que ele se refere: garantir que uma classe tenha apenas uma instância, e providenciar um ponto global de acesso à ela. A solução: define-se um método denominado *Instance* na classe cuja instância deve ser única em todo o sistema. Esse método é o responsável por criar uma única instância da classe, e fornecer acesso à ela para o restante do sistema. Consequências: acesso controlado à instância da classe, permite refinamento de operações e representação, permite um número variado de instâncias, oferece maior flexibilidade do que operações de classe.

2.1.1.6 Frameworks

Frameworks são conjuntos de classes que implementam, em conjunto, a funcionalidade básica de um domínio ou técnica de negócios. Geralmente os desenvolvedores utilizam *frameworks* como base para a construção de aplicações.

Existem *frameworks* que implementam os componentes básicos de GUIs (Graphical User Interface), que implementam lógica de persistência, *frameworks* para seguros, recursos humanos, manufatura, bancos e comércio eletrônico. O reuso de *frameworks* representa um alto nível de reutilização no nível de domínio.

Embora o nível de reuso seja grande quando se utiliza *frameworks*, uma série de desvantagens algumas vezes dificulta seu uso. Gasta-se bastante tempo estudando-se e aprendendo-se a lógica de um *framework*, devido à sua grande complexidade. Além disso, *frameworks* diferentes dificilmente trabalham juntos, a não ser que venham de um mesmo fornecedor.

2.1.1.7 Assets

De acordo com o OMG (Object Management Group), um *asset* é uma coleção de artefatos reusáveis que fornecem uma solução para um problema em um dado contexto (OMG 2005). Esses artefatos, como já visto anteriormente, podem ser quaisquer artefatos utilizados no processo de desenvolvimento, tais como documentação, casos de uso, modelos, código fonte, casos de testes, etc.

Um *asset* pode ter vários pontos de variabilidade, que permitem ao usuário customizá-lo. Um ponto de variabilidade é uma localização no *asset* que pode ter um valor fornecido ou customizado pelo consumidor do *asset*. Além disso, possui regras de uso, que são instruções que descrevem como (e em que contexto) o *asset* deve ser usado.

Existem três dimensões que descrevem *assets* reusáveis: granularidade, variabilidade e articulação. Essas dimensões são descritas na Figura 2.1 (OMG 2005).

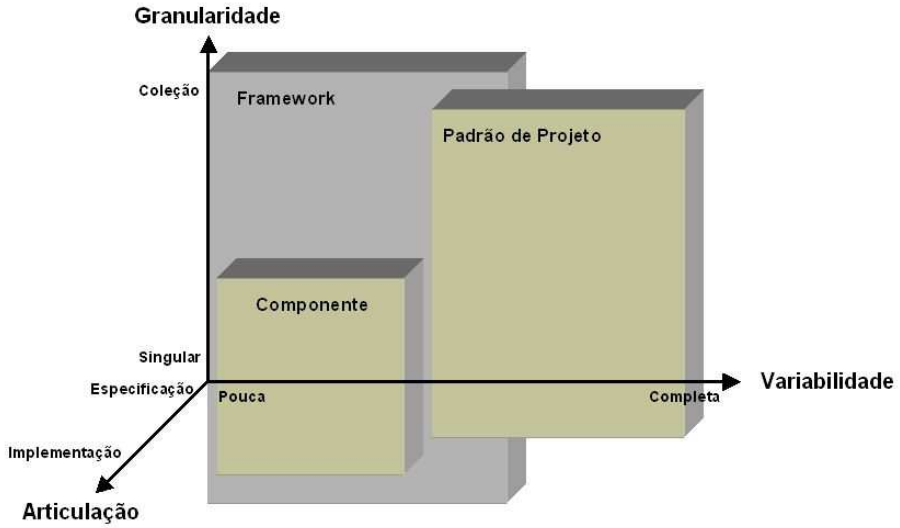

Figura 2.1: Dimensões de *assets* reusáveis

- Granularidade: a Granularidade de um *asset* descreve a quantidade de problemas ou soluções alternativas são cobertas pelo *asset*. *Assets* mais simples oferecem um solução singular para um problema único e bem definido. Conforme a granularidade aumenta o *asset* resolve múltiplos problemas e/ou pode oferecer soluções alternativas para esses problemas. Em geral, com o aumento da granularidade aumenta também o tamanho e a complexidade do *asset*.

- Variabilidade: a variabilidade refere-se a quão customizável um *asset* pode ser. Em um extremo um *asset* pode ser invariável, ou seja, não pode ser alterado de forma alguma. Esse é o caso de *assets* que são componentes binários. Eles são chamados de *assets caixa-preta*. Da mesma maneira, existem os *assets caixa-branca*. Esses *assets* são criados com a expectativa que seus consumidores irão alterar e editar sua implementação. *Assets caixa-branca* incluem tipicamente artefatos de desenvolvimento. Há outras duas variações que estão no meio-termo, são os *assets caixa-clara* e os *assets caixa-cinza*. *Assets caixa-clara* apresentam detalhes de implementação (para que o consumidor possa utilizá-lo mais adequadamente), porém não podem ser modificados. *Assets caixa-cinza* apresentam e permitem modificações somente em um subconjunto de artefatos do *asset*, geralmente através de parâmetros.

- Articulação: essa dimensão descreve o grau de completude dos artefatos em prover a solução. *Assets* cujos artefatos especificam a solução mas não fornecem a solução tem um baixo grau de articulação. Já *assets* cujos artefatos especificam e fornecem a solução, juntamente com os documentos de suporte (requisitos, casos de uso, testes, etc), têm um alto grau de articulação.

Um *asset*, para ser disponibilizado para reuso deve ser documentado e catalogado. Para isso, a OMG propôs a especificação RAS (Reusable Asset Specification). Um *asset* RAS tem quatro seções básicas: Classificação (atributo para pesquisa e navegação),

Solução (conjunto de artefatos que implementam a solução), Uso (instruções de como aplicar e utilizar o *asset*) e *Assets* Relacionados (definem a relação do *asset* com outros *assets*).

2.1.2 Problemas na Adoção do Reuso

O reuso é uma técnica que já foi considerada como a grande "bala de prata" para se resolver os problemas da Engenharia de Software (COX 90). No entanto, uma série de fatores faz com que o reuso não seja tão utilizado quanto poderia.

Na tabela 2.1, fornecida por Yongbeom e Stohr (YON 98), são descritos os principais problemas e dificuldades na adoção do reuso.

Categoria	*Aspectos Pesquisados*	*Impedimentos do reuso*
Gerais	Definição de 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

Tabela 2.1: Problemas e dificuldades na adoção do reuso

2.1.3 Benefícios na Adoção Reuso

Os principais benefícios do reuso são descritos a seguir (ALM 2007).

- Qualidade: conforme um artefato é reusado, a quantidade de erros que ele possui tende a diminuir.
- Produtividade: os membros da equipe precisam produzir menos artefatos, logo conseguem produzir mais em um intervalo menor de tempo.
- Confiabilidade: quanto mais um artefato é reusado, mais ele é testado, aumentando, assim, a confiança que se pode depositar no artefato.

- Tempo de desenvolvimento: com uma série de componentes já prontos, o tempo de desenvolvimento do sistema diminui.
- Documentação: quanto menos código novo for criado, menos documentação deverá ser produzida. Além disso, com o reuso, há mais tempo para melhorar a documentação já existente.
- Custos de manutenção: com o reuso, menos erros ocorrem, melhor qualidade é obtidade, e a necessidade de manutenção se torna menor. Se houver necessidade de alteração de código, com o reuso, este código tende a estar mais concentrado e as alterações podem ser mais pontuais.
- Tamanho das equipes: grandes equipes de desenvolvimento tendem a ter a comunicação dificultada. Muitas vezes aumenta-se o número de integrantes de uma equipe com o intuito de aumentar a produtividade, mas isso nem sempre ocorre na proporção esperada. Com o reuso de componentes de software, os sistemas poderiam ser desenvolvidos por equipes menores, facilitando assim a comunicação entre os integrantes.

3 PROCESSO DE DESENVOLVIMENTO DE IU E O REUSO

Neste capítulo, serão apresentadas as principais etapas no desenvolvimento de sistemas, considerando-se os diversos níveis de abstração que compõem esse processo. Será apresentado também o conceito de **árvore de reificação**, essencial para o entendimento desse trabalho.

3.1 Desenvolvimento de Sistemas Dirigido por Casos de Uso

Desenvolvimento iterativo é uma forma de se construir sistemas em que o ciclo de desenvolvimento é composto de várias iterações em sequência (LAR 2003). Nesse processo, cada iteração é composta pelas atividades de análise de requisitos, modelagem, codificação e teste. Dessa forma, ao fim de cada iteração obtém-se uma parte completa do sistema, já testada e integrada. O objetivo é que ao final de todas as iterações, tenha-se o sistema completo.

De acordo com Bezerra (BEZ 2002), a cada ciclo de desenvolvimento um novo subconjunto de requisitos é considerado para ser desenvolvido, o que produz um incremento do sistema, que contém extensões e refinamentos sobre o incremento anterior. Na Figura 3.1, retirada do livro de Bezerra (BEZ 2002), pode-se ver o ciclo de desenvolvimento de um sistema composto por ciclos menores (*releases*).

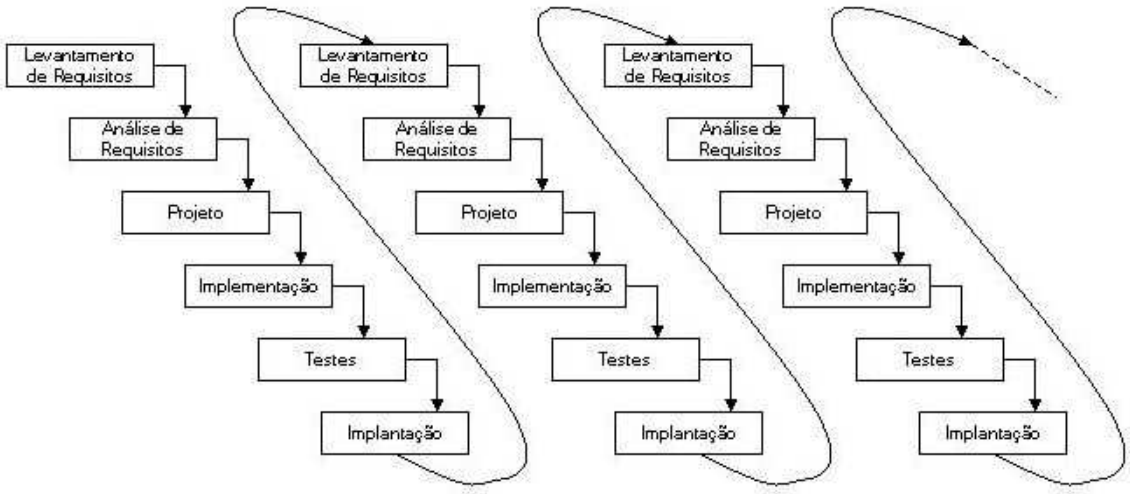

Figura 3.1: Modelo de desenvolvimento iterativo e incremental

Com base no modelo de desenvolvimento iterativo e incremental, Ivar Jacobson propôs a organização dos ciclos de desenvolvimento em casos de uso (JAC 92). Essa proposta

originou o termo **desenvolvimento dirigido a casos de uso** (*use case driven development*).

Nessa proposta, um subconjunto de casos de uso (ou de versões simplificadas de casos de uso) é atribuído a cada ciclo para ser desenvolvido. Os casos de uso são agrupados, geralmente, considerando a prioridade (para o cliente) e o risco de desenvolvimento. Além de serem usados para capturar requisitos funcionais e definir ciclos de desenvolvimento, nessa abordagem os casos de uso podem desempenhar um papel mais central e signiticativo no processo de desenvolvimento

Um exemplo de processo de desenvolvimento iterativo é o **Processo Unificado** (PU) (KRU 2000). O PU é um processo de desenvolvimento dirigido a casos de uso. Isso significa que os casos de uso são a base para todo o processo de desenvolvimento, pois dirigem atividades como análise, modelagem, codificação e testes. Essa habilidade dos casos de uso de unificar as atividades de desenvolvimento os torna um poderoso instrumento para o planejamento e acompanhamento de projetos de desenvolvimento de software.

3.2 Casos de Uso na UML: Conceitos e Elementos

Um dos principais conceitos para se entender casos de uso é o conceito de ator. Dessa forma, antes de discutir casos de uso propriamente ditos, será apresentada a definição de ator na UML (OMG 2007).

3.2.1 Atores

Um ator, na UML, é todo e qualquer elemento que não faz parte do sistema e que interage com o sistema. De acordo com Craig Larman (LAR 2001), um ator é uma entidade externa ao sistema, que de alguma maneira participa da história do caso de uso.

Atores podem representar pessoas, os usuários do sistema ou outros agentes externos que interagem com o sistema, como outros sistemas de software ou dispositivos de hardware. Geralmente, os atores de um sistema são mais facilmente identificáveis do que os casos de uso do sistema.

Na UML, um ator é representado pela Figura 3.2.

Figura 3.2: Ator na UML

Um ator pode se relacionar com outro ator através de relacionamentos de generalização/especialização. Assim, um ator com características mais comuns pode ser especializado por outro com características mais definidas. Em um sistema de compras, por exemplo, pode haver um ator Usuário que é especializado pelos atores Comprador e Vendedor. Na UML, esse relacionamento é representado pela Figura 3.3.

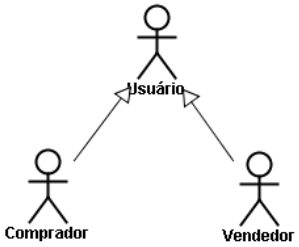

Figura 3.3: Representação do relacionamento generalização/especialização entre atores na UML

3.2.2 Casos de Uso

Casos de uso são uma forma de se especificar as funcionalidades requeridas de um sistema (OMG 2007). A descrição de um sistema pode ser composta de vários casos de uso, cada um prevendo uma funcionalidade e uma série de cenários onde o sistema deve/pode ser executado.

São documentos essencialmente textuais, podendo também ser representados através de diagramas de fluxo, diagramas de seqüência, redes de Petri ou até mesmo linguagens de programação. Casos de uso contêm narrativas que mostram como os atores interagem com o sistema para atingir um objetivo. As narrativas de um caso de uso descrevem a interação entre o sistema e um ou mais atores, de modo que o objetivo do(s) ator(es) seja alcançado (COC 2000). Um exemplo de caso de uso textual pode ser visto na Tabela 3.1.

Título: Visualizar pedido de compra pela Internet
Ator: Cliente
Descrição: Cliente realiza a visualização de um pedido
Pré-condições: Cliente deve ter pelo menos um pedido cadastrado no sistema.

	Ator		**Sistema**
1	Clica no botão "Meus Pedidos"		
		2	Solicita informações de usuário e senha do cliente
3	Digita informações de usuário e senha		
		4	Verifica corretude das informações de usuário e senha fornecidas
		5	Lista os pedidos do cliente
6	Seleciona o pedido que deseja visualizar		
7	Clica no botão "Visualizar Pedido"		
		8	Mostra informações do pedido selecionado

Tabela 3.1: Caso de uso de visualização de pedido

Um caso de uso na UML é representado pela Figura 3.4.

Figura 3.4: Representação gráfica de um caso de uso na UML

Casos de uso podem se relacionar entre si. Os tipos de relacionamentos possíveis entre casos de uso são:

- Inclusão: quando um caso de uso contém o comportamento definido em outro caso de uso. Esse relacionamento é representado pela Figura 3.5.

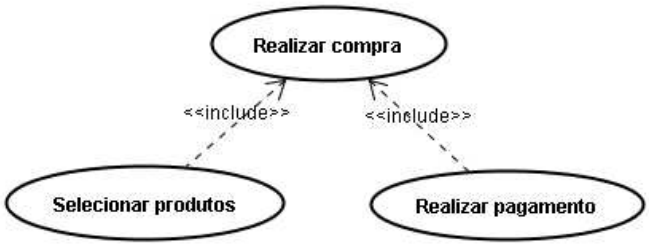

Figura 3.5: Relacionamento de inclusão entre casos de uso

- Extensão: identifica um ponto no comportamento do caso de uso onde ele pode ser estendido pelo comportamento de outro caso de uso. Esse relacionamento é representado pela Figura 3.6.

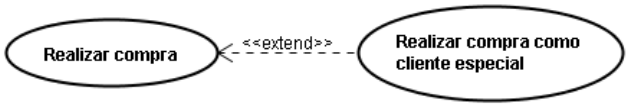

Figura 3.6: Relacionamento de extensão entre casos de uso

- Generalização/especialização: quando um caso de uso base possui diferentes especializações. Esse relacionamento adiciona diferentes comportamentos ao caso de uso base. É representado pela Figura 3.7.

Figura 3.7: Relacionamento de generalização/especialização entre casos de uso

Como se pôde observar nos relacionamentos de extensão, inclusão e generalização/especialização entre casos de uso e o relacionamento de generalização/especialização entre

atores, a UML fornece embutido no conceito desses relacionamentos, possibilidades de aplicar técnicas de reuso já no nível de análise do sistema. No relacionamento de generalização/especialização, por exemplo, o analista pode reutilizar o mesmo caso de uso base em várias situações.

3.3 Projeto de Interface de Usuário (IU) no Desenvolvimento de Sistemas

Segundo Cockburn (COC 2000), casos de uso são o ponto de conexão entre os requisitos do sistema. Além disso, os casos de uso são muito úteis nas etapas de modelagem de IU e testes. A Figura 3.8, extraída do livro de Cockburn (COC 2000), exemplifica essa visão de como os casos de uso atuam entre partes essenciais do sistema.

Figura 3.8: Modelo de requisitos

No PU, o processo de definição de IU deve acontecer após a análise dos requisitos funcionais, que na UML são representados pelos casos de uso. A partir da descrição, presente no caso de uso, constrói-se um protótipo da IU. Dessa forma, pode-se validar os requisitos funcionais e verificar se a IU está fornecendo todos os elementos e passos necessários para que o objetivo do caso de uso seja alcançado.

Para se definir o protótipo, um dos elementos mais importantes a ser considerado é o ambiente de interação. Nesta etapa analisam-se as tecnologias disponíveis (sem que elas necessariamente já sejam definidas) para a implementação do sistema e principalmente o ambiente onde ele vai ser executado (se *desktop*, *web*, ou *mobile*, por exemplo.) A partir daí, o protótipo é construído e refinado de acordo com a necessidade até que seja aceito e aprovado pelos usuários.

Uma vez que se tenha os casos de uso definidos, refinados e com descrições apuradas, os protótipos prontos e aprovados, é hora, então, de se implementar a solução como um todo.

Apesar dessa abordagem parecer bastante clara e definida na teoria, na prática o papel dos casos de uso no desenvolvimento e especificação de IUs ainda é uma questão em

aberto. Existem muitas dificuldades e falta de métodos e técnicas para se obter UIs de boa qualidade e usabilidade a partir de casos de uso. Um dos grandes problemas é que a engenharia de requisitos tem usualmente voltado sua atenção para os requisitos funcionais, dando pouco ou nenhum enfoque para os requisitos de usabilidade ou interação. Dessa forma, as abordagens tradicionais incitam os analistas a definir IUs 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 (PIM 2000).

3.4 Modelos de Especificação de IU

Modelos de IU descrevem, de forma abstrata, a IU a ser criada. Modelos podem ser definidos como uma descrição formal, declarativa, e livre de implementação (EIS 2000). Os modelos descrevem a implementação da interface utilizando-se de componentes abstratos, que mais tarde serão mapeados para componentes de interface da tecnologia alvo da aplicação.

Uma IU é descrita através de um conjunto de **objetos de interação**, ou IO (*Interaction Objects*). Um IO é qualquer elemento que permita aos usuários de uma aplicação visualizar ou manipular informações ou realizar uma tarefa interativa.

Existem dois tipos de IOs (VAN 93):

- *Concrete Interaction Object* (CIO). Qualquer entidade da IU que pode ser percebida pelo usuário (de modo geral, são os objetos gráficos da aplicação, também conhecidos como *widgets* ou *windows*). Um CIO contém as características do ambiente de interação (se Windows XP, Linux, mobile, etc) e as restrições (*constraints*) que devem ser respeitadas quando um usuário manipula um objeto.

- *Abstract Interaction Object* (AIO). Visão abstrata do CIO. Descreve as características físicas do CIO sem considerar o ambiente de interação. Um AIO não tem aparência gráfica, mas é conectado a zero ou mais CIOs em diferentes ambientes de interação, com diferentes nomes e apresentações.

De modo geral, a relação entre os objetos de interação é representada pela Figura 3.9.

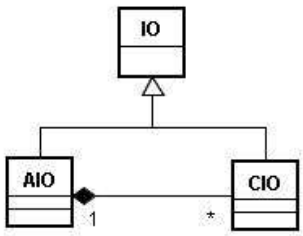

Figura 3.9: Relação entre objetos de interação

Existem, ainda, três tipos de representação de IOs:

- Os que representam espaços de interação: ambientes onde devem acontecer as interações do usuário com o sistema.

- Os que representam mecanismos de interação: mecanismo através do qual acontece a interação do usuário com o sistema.

- Os que representam elementos de interação: elementos que compõem o mecanismo de interação. São os menores objetos de interação com os quais um usuário pode interagir, sendo, portanto, indivisíveis.

De modo geral, elementos de interação podem ser combinados de forma a criar mecanismos de interação. Um mecanismo de interação, em um espaço de interação, proporciona diferentes formas de se realizar a mesma interação. Como exemplo, considere-se o conjunto de elementos de interação de caixa de texto, botão e calendário. Juntos, esses elementos são organizados em um espaço de interação (ambiente *desktop*, ambiente *web*) e implementam um mecanismo de interação para o preenchimento de data pelo usuário.

3.4.1 Modelo CRF para Desenvolvimento de IU

Conforme visto na seção anterior, uma IU é uma coleção de objetos de interação que se organizam em uma hierarquia. O *Cameleon Reference Framework* (CRF) (CAL 2003) propõe uma forma de organizar os objetos que compõem a IU em diferentes níveis de abstração. Nesse modelo, são definidos seis passos para de se descrever IUs, que podem ser vistos na Figura 3.10.

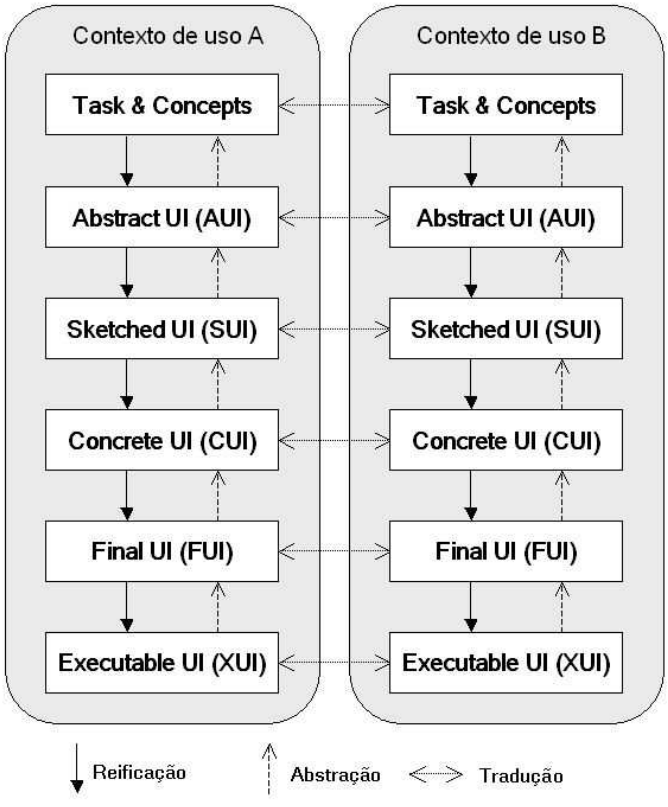

Figura 3.10: Cameleon Reference Framework

- *Task and Concepts*: a tarefa a ser executada e os conceitos orientados ao domínio requeridos para o desempenho dessa tarefa são obtidos dos padrões de caso de uso e padrões de interação.

- *Abstract User Interface* (AUI): uma abstração de uma CUI (definida logo abaixo), 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 no ambiente de interação. É basicamente uma descrição abstrata da IU definida no nível anterior.

- *Sketched User Interface* (SUI): conjunto de artefatos com o esboço (desenho, *screenshot*, etc) e uma descrição textual da IU sendo modelada. Deve haver uma e somente uma SUI por modelo de IU. Todos os demais artefatos são formalizações e detalhamentos da IU descrita na SUI.

- *Concrete User Interface* (CUI): é um modelo de IU que permite a especificação da aparência e do comportamento da IU através dos objetos de interação que podem ser percebidos pelo usuário (botões, caixas de texto, etc.). Apesar de ser independente da plataforma computacional e linguagem de programação, uma CUI depende do ambiente de interação.

- *Final User Interface* (FUI): uma IU descrita em linguagem de programação, ou seja, são arquivos de texto com o código-fonte da linguagem que implementa a IU. Uma FUI pode incluir outras FUIs e invocar chamadas a XUIs.

- *Executable User Interface* (XUI): é um programa de computador executável, em uma determinada plataforma, que renderiza a IU. Geralmente, artefatos reusáveis na forma de XUIs são componentes de IU já implementados.

O CRF define o modelo de IU como a base sob a qual os artefatos reusáveis são identificados, produzidos, consumidos e organizados. O CRF define ainda duas maneiras de se obter os passos de desenvolvimento: através de tranformações horizontais e verticais.

As transformações verticais se referem à transformação de um passo de desenvolvimento (um modelo de IU) em outro. Transformar um modelo de IU de uma forma mais abstrata para uma forma mais concreta é chamado processo de reificação. A transformação de um objeto mais concreto em outro mais abstrato é o 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).

3.5 Abordagem de Reuso Proposta

A abordagem de reuso proposta por Augusto (MOR 2007) foi inspirada no desenvolvimento de IUs para aplicações multi-contexto, o CRF (*Cameleon Reference Framework*), citado na seção anterior. 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 numa **árvore de reificação**.

O reuso deve acontecer através da especificação de modelos de reificação de padrões de interação e padrões de caso de uso. A ferramenta proposta neste trabalho foca seus esforços nos padrões de caso de uso.

Um padrão de caso de uso contém uma hierarquia de artefatos reusáveis que implementam, em diversos níveis de detalhe, as soluções descritas pelo padrão de caso de uso.

Um caso de uso pode ter vários modelos de IU. 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. O conjunto de artefatos que

compõem a hierarquia, incluindo o padrão de caso de uso, é chamado **padrão de caso de uso reificado** (RUCP).

No RUCP, uma SUI constitui a apresentação do padrão de caso de uso (é o protótipo). Os outros elementos são os níveis de reificação deste protótipo. Como um padrão de caso de uso deve ser escrito de forma independente de tecnologia, ele pode conter SUIs e CUIs que o apresentem em um ambiente *web* de interação, em um ambiente gráfico *desktop* e assim por diante, para Para cada ambiente pode haver FUIs e XUIs que implementem o padrão de caso de uso em diferentes linguagens de programação e/ou arquiteturas.

É principalmente nessa visão hierárquica dos artefatos que compõem um RUCP que a ferramenta proposta nesse trabalho se baseia. Ela tem como objetivo oferecer a catalogação e busca de diferentes tipos de artefatos, de modo a propiciar reuso de IUs em diferentes níveis de abstração.

3.6 Soluções Existentes

A adoção de técnicas de reuso permite, a longo prazo, obter-se ganhos, principalmente, em qualidade e produtividade. Mudanças nas metodologias de trabalho são necessárias para se inserir no dia-a-dia a busca por soluções existentes. Para que isso ocorra, essas soluções tem de estar catalogadas e organizadas em um repositório de acesso geral.

A organização dos artefatos reusáveis em hierarquias, como visto nas seções anteriores, permite que o reuso ocorra em diversos níveis da aplicação. No entanto, poucas são as ferramentas que oferecem suporte a reuso em mais de um nível. A maior parte das ferramentas encontradas durante o processo de pesquisa deste trabalho oferece busca e catalogação de componentes e trechos de código fonte, não oferecendo nenhum tipo de catalogação e organização de artefatos em mais de um nível.

Foram encontradas diversas ferramentas de suporte a reuso durante o processo de pesquisa deste trabalho. Dessas soluções, três foram selecionadas para serem descritas a seguir:

- Koders (KOD 2007): repositório *web* de código-fonte. Nele os usuários podem tanto hospedar quanto baixar trechos de código-fonte e algoritmos nas mais diversas linguagens. Oferece extensões que podem ser adicionadas a IDEs (Integrated Development Enviroment). Não oferece, porém, opções de *upload* de código.

- Mero Base Component Finder (MER 2007): repositório, também *web*, de código fonte e de componentes. Nesse repositório, a busca de componentes acontece de três formas: por nome, por *substring* presente no código-fonte, por procedimentos, e por objetos e classes. Diferentemente da ferramenta Koders, essa ferramenta oferece *upload* de código e de componentes.

- Odissey (UFR 2007): ferramenta *desktop* desenvolvida pela Equipe de Reutilização de Software da UFRJ. Oferece suporte a reutilização de software baseada em modelos de domínio. Criam-se domínios de aplicação que ficam disponíveis para serem buscados e reutilizados em projetos futuros. Fornece métodos, ferramentas e procedimentos para a especificação de modelos e aplicações do domínio, além de ferramentas para aquisição de conhecimento, editor de diagramas orientados a objeto, navegador e ferramenta de documentação. Possui uma abordagem diferenciada, se comparada às abordagens encontradas na maior parte das ferramentas. Enquanto as outras ferramentas encontradas focam em componentes, algoritmos e trechos de código fonte, essa ferramenta foca em um nível mais abstrato.

O grande problema dessas ferramentas, porém, é que, excetuando a ferramenta Odissey, elas não oferecem *download* da ferramenta em si. Seus repositórios estão na *web* e existe o risco de eles sairem do ar a qualquer momento. Outra limitação é com relação à pesquisa de artefatos: como os repositórios são únicos, há muitos artefatos cadastrados, o que dificulta ainda mais a pesquisa. Além disso, nenhuma das ferramentas encontradas oferecem suporte a reuso de IU.

Assim, o processo de reuso como um todo carece de ferramentas que sejam, ao mesmo tempo, abrangentes e de fácil utilização, de forma a tornar a busca por soluções existentes e o efetivo reuso dessas soluções mais atrativo do que a construção de uma solução partindo do zero.

Pensando em todas essas limitações encontradas nas ferramentas já existentes, este trabalho propõe-se a apresentar uma ferramenta com suporte a reuso de artefatos de IU, em diversos níveis de aplicação.

4 FERRAMENTA PROPOSTA: REUSE

Neste capítulo será apresentado o protótipo da ferramenta, além dos relacionamentos existentes entre os artefatos que compõem o CRF e de que forma eles influenciaram na construção da ferramenta ReUse.

4.1 Organização dos Artefatos

Como visto na seção anterior, existe uma série de ferramentas que são repositórios de código e de componentes disponíveis na *web*. Não foi encontrada nenhuma, no entanto, que atendesse abordagens mais abrangentes de reuso. Assim, propõe-se neste trabalho, a apresentação de uma ferramenta que oferece suporte a reuso de software, mais especificamente IU em diferentes níveis de abstração.

A hierarquia de artefatos reusáveis que compõe o RUCP foi segmentada em três grandes grupos, onde os artefatos são organizados. Essa divisão foi proposta para simplificar a organização e principalmente a busca dos artefatos para o reuso. Um ou mais passos do CRF foi designado para cada um desses três grupos de artefatos. Esses três grupos de artefatos são:

- **Casos de Uso ou Padrões de Casos de Uso**: esse é o nível mais abstrato da árvore. Nele se encontram os artefatos que definem, de fato, o que deve ser implementado. Esse grupo contém essencialmente os casos de uso ou padrões de casos de uso. Cada caso de uso relaciona-se com pelo menos um artefato de especificação.

- **Artefatos de Especificação**: são os artefatos referentes ao modelo da IU a que o caso de uso referencia. São artefatos mais refinados, pois contêm esboços e descrições mais detalhadas da IU proposta pelo caso de uso. Os passos do CRF que compõem esse grupo de artefatos são: AUIs, SUIs e CUIs. Uma AUI pode ter diversas SUIs assim como uma SUI pode conter várias CUIs.

- **Artefatos de Implementação**: são os artefatos mais "baixo nível"da árvore de reificação. Os passos do CRF a que esse grupo de artefatos se refere são o FUI e o XUI. Nesse grupo são encontradas as implementações das IUs definidas nos passos anteriores. Assim, um artefato do nível de especificação pode se relacionar com inúmeros artefatos do nível de implementação. Basta, para isso, pensarmos que uma mesma IU pode ser implementada em mais de uma linguagem de programação e em diferentes plataformas, ou ambientes de interação.

A árvore de reificação e os relacionamentos descritos acima podem ser vistos através da Figura 4.1.

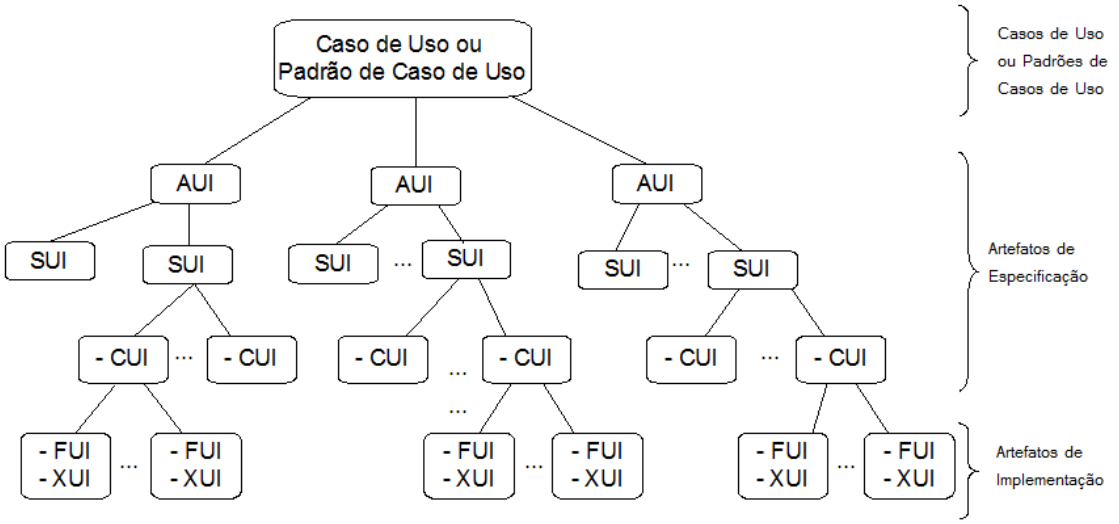

Figura 4.1: Árvore de reificação

4.2 ReUse: Ferramenta de Suporte a Reuso

A ferramenta de suporte a reuso apresentada neste trabalho tem como base as etapas de desenvolvimento de IUs propostas pelo CRF e reunidas em três grupos, apresentados na seção anterior. O principal objetivo é oferecer de forma livre e gratuita uma ferramenta que pode ser usada tanto internamente dentro de uma organização, quanto diretamente na *web*, como um repositório universal.

A ReUse é uma ferramenta *web*, desenvolvida em linguagem PHP (PHP 2007), sobre um *framework*, o phpWAFr (REZ 2007) e utilizando aplicativo de banco de dados MySQL (MYS 2007). As escolhas referentes à implementação da ferramenta foram feitas com o intuito de facilitar o processo de desenvolvimento, através da escolha de um *framework*, que é uma das formas de reuso de *software* citada nesse trabalho.

A idéia básica é fazer com que diferentes artefatos sejam submetidos através da ferramenta e analisados por um administrador, que decidirá se os artefatos propostos tem realmente possibilidade de reuso. Além disso, qualquer usuário deve ser capaz de realizar buscas de artefatos, em quaisquer níveis.

A catalogação e organização dos artefatos, do ponto de vista da ferramenta, é apresentada a seguir, sendo seus critérios de busca e classificação explicados e devidamente justificados.

4.2.1 Artefato Caso de Uso

Este tipo de artefato, como apresentado anteriormente, engloba padrões de casos de uso e casos de uso.

O catálogo desse tipo de artefato é dividido em cinco pequenas etapas ou abas, enumeradas e descritas a seguir na ordem em que aparecem e pelo título correspondente:

1. **Aba Geral**: contém as informações essenciais referentes ao artefato. Essas informações são:

   - Pacote: refere-se ao pacote a que o caso de uso pertence. Imaginando um projeto de software, o pacote seria uma subdivisão dentro do sistema. Em um sistema de folha de pagamento, por exemplo, haveria os casos de uso que

definem o cadastro dos usuários, os casos de uso que definem os cálculos da folha de pagamento, etc. Cada subgrupo desses é representado por um pacote.

- Nome: é o nome do caso de uso.
- Data de Criação: data em que o artefato foi criado. Pode-se considerar a data real em que ele foi criado ou então a data em que ele foi inserido no sistema.
- Data Última Alteração: é a data da última alteração realizada no artefato.
- Descrição: é a descrição do artefato. É um dos pontos chave da ferramenta, pois uma busca com bons resultados só será possível se os campos, principalmente da descrição, forem bem definidos.
- Versão: versão do artefato cadastrado no repositório.
- Pré-condições: são as pré-condições do caso de uso.
- Pós-Condições: são as pós-condições do caso de uso.
- Atores: são os atores que participam do caso de uso.
- Autor: correponde ao usuário logado no sistema que inseriu o caso de uso na base de dados.
- Caso de uso (arquivo): é o arquivo do caso de uso completo, a ser submetido e colocado à disposição para reuso posterior no sistema.

2. **Aba Associar Coleções**: esse elemento do catálogo do caso de uso aparece na tentativa agrupar os artefatos de uma forma horizontal. A abordagem da árvore de reificação leva a entender a relação entre os artefatos de uma maneira vertical. Conhece-se um caso de uso e a partir dele conhecem-se os artefatos que compõem o RUCP. Nada se sabe, porém, do relacionamento dos artefatos casos de uso, artefatos de especificação e artefatos de implementação uns com os outros. Assim, esse critério pode se referir a uma plataforma, um framework, um ambiente de interação, ou até mesmo uma aplicação, que tem seus artefatos cadastrados na ferramenta e disponibilizados para reuso.

3. **Aba Associar Artefato de Especificação**: possibilidade de se adicionar artefatos de especificação, previamente cadastrados, ao caso de uso. Essa associação é o elo de ligação entre o nível mais abstrato da implementação de um sistema e os níveis intermediários, com mais detalhes de implementação, porém, sem conter a implementação propriamente dita.

4. **Aba Artefatos Relacionados**: nesta aba são apresentados todos os artefatos de especificação relacionados com o caso de uso, de acordo com a Figura 4.1. A partir da lista, pode-se visualizar os detalhes de cada artefato relacionado ao caso de uso.

5. **Aba Comentários**: nesta aba, os usuários podem adicionar comentários referentes ao artefato. Se um artefato foi adicionado com pendências de informação, seu próprio dono pode inserir um comentário como aviso para os outros usuários que forem reutilizar o artefato. Além disso, os outros usuários que reusarem o artefato podem adicionar suas opiniões, reportar erros ou até mesmo fazer sugestões de melhorias, facilitando, assim, o seu refinamento.

Um exemplo da tela descrita pelos ítens acima, com todas as abas e as informações essenciais, pode ser visto na Figura 4.2

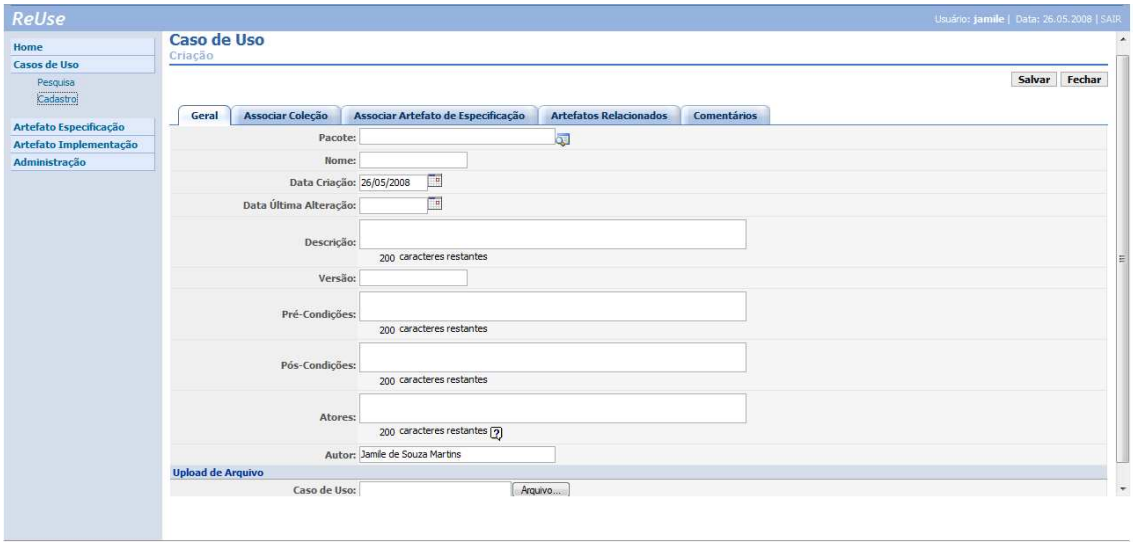

Figura 4.2: Tela de cadastro de caso de uso

Além das opções de catalogação, outra etapa essencial em um ambiente de suporte a reuso é a pesquisa dos artefatos. Os critérios para a pesquisa de artefatos são um dos pontos de maior importância em uma ferramenta para reuso. Aliados a boas descrições, podem tornar o processo de busca por uma solução existente efetivamente mais rápido do que a construção de uma nova solução ou de soluções semelhantes para o mesmo problema.

Para que essas expectativas sejam atendidas, no caso da pesquisa por casos de uso, os critérios de pesquisa escolhidos foram os seguintes: identificador (corresponde ao identificador do artefato inserido na base de dados), nome, descrição, versão, atores, pré-condições, pós-condições, pacote, coleção, data de criação, data da última alteração e autor. Um exemplo de resultado de pesquisa por um artefato caso de uso pode ser visto na Figura 4.3.

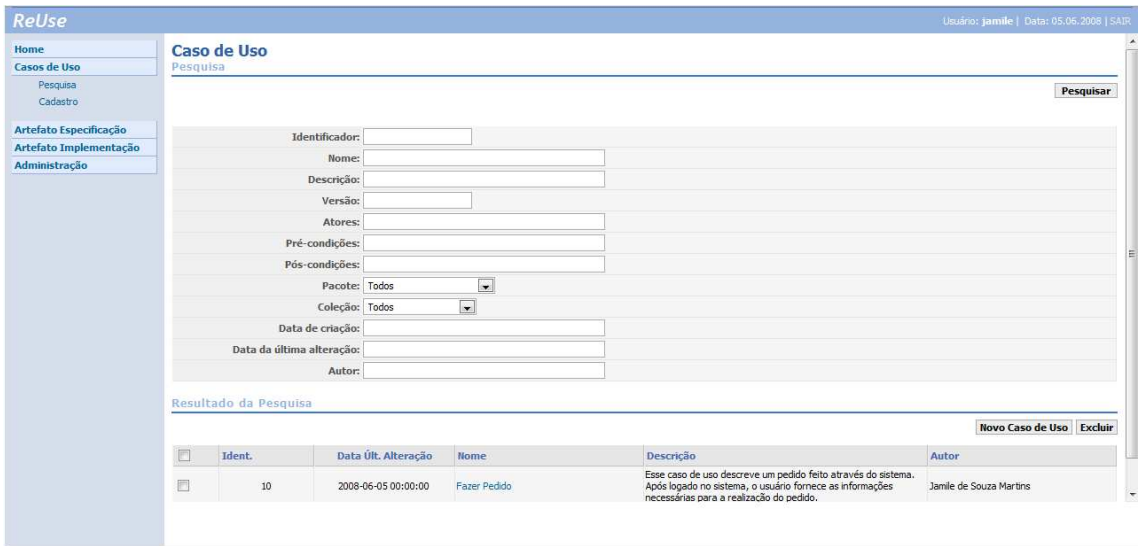

Figura 4.3: Tela de pesquisa de caso de uso

Quaisquer dos filtros podem ser preenchidos, sem restrição nenhuma dos dados informados. Os campos cujos valores são *strings* são pesquisados na base de dados através de

substrings dentro dos campos existentes.

4.2.2 Artefato de Especificação

Este grupo de artefatos se refere às etapas AUIs, SUIs e CUIs do CRF. Um artefato de especificação está relacionado a no máximo um artefato de nível mais abstrato, ou seja, a um caso de uso.

O catálogo desse tipo de artefato é dividido em seis pequenas etapas ou abas, enumeradas e descritas a seguir na ordem em que aparecem e pelo título correspondente:

1. **Aba Geral**: da mesma forma que nos casos de uso, essa aba contém as informações essenciais a respeito do artefato. Os campos que a compõem são os seguintes:
   - Nome: corresponde ao nome do artefato a ser cadastrado.
   - Data de criação: data em que o artefato foi criado ou inserido no repositório.
   - Data da última edição: data da última alteração realizada no artefato.
   - Descrição: corresponde à descrição do artefato. Nesse tipo de artefato esse é um dado particularmente importante, já que ele engloba mais de um passo do CRF.
   - Versão: versão do artefato cadastrado no repositório.
   - Tipo: esse é um dado que diferencia os artefatos de especificação dos casos de uso. Como esse grupo de artefatos tem associado a si mais de uma etapa prevista no CRF, é nesse momento que se diz qual o tipo do artefato a ser inserido: se uma SUI, AUI ou CUI.
   - Autor: correponde ao usuário logado no sistema que inseriu o caso de uso na base de dados.
   - Artefato de especificação (arquivo): é o artefato de especificação propriamente dito. Aqui pode-se ter: no caso de uma SUI, uma imagem; no caso de uma AUI, uma descrição abstrata de uma IU; ou, no caso de uma CUI, uma descrição concreta da IU, levando em consideração os elementos disponíveis para sua criação e o ambiente de interação a que a IU pertence.
2. **Aba Associar Coleção**: refere-se à mesma tentativa presente nos artefatos casos de uso, de oferecer um agrupamento horizontal entre os artefatos, indicando se eles pertencem a uma mesma aplicação, a um *framework*, etc.
3. **Aba Associar Artefato de Especificação**: como visto na Figura 4.1, um artefato de especificação pode estar associado a outros artefatos de especificação. Uma AUI pode ser possuir váriias SUIs, que por sua vez, podem ser compostas de várias CUIs. Essa aba possibilita indicar esse relacionamento, de um artefato de especificação com outros artefatos de especificação previamente cadastratos.
4. **Aba Associar Artefato de Implementação**: ainda de acordo com a árvore de reificação apresentada na Figura 4.1, um artefato de especificação pode estar associado a um ou vários artefatos de implementação, bastando imaginar que uma IU pode ter implementações em diferentes linguagens de programação, plataformas, etc.
5. **Aba Artefatos Relacionados**: aqui são listados todos os casos de uso, artefatos de especificação e artefatos de implementação relacionados com o artefato de especificação cadastrado.

6. **Aba Comentários**: os comentários dos artefatos de especificação têm o mesmo objetivo que nos artefatos casos de uso. Anotações podem ser feitas sobre os artefatos, possibilitando, assim, melhorias e refinamentos em versões posteriores.

A tela de inserção/edição de artefatos de especificação pode ser vista na Figura 4.4

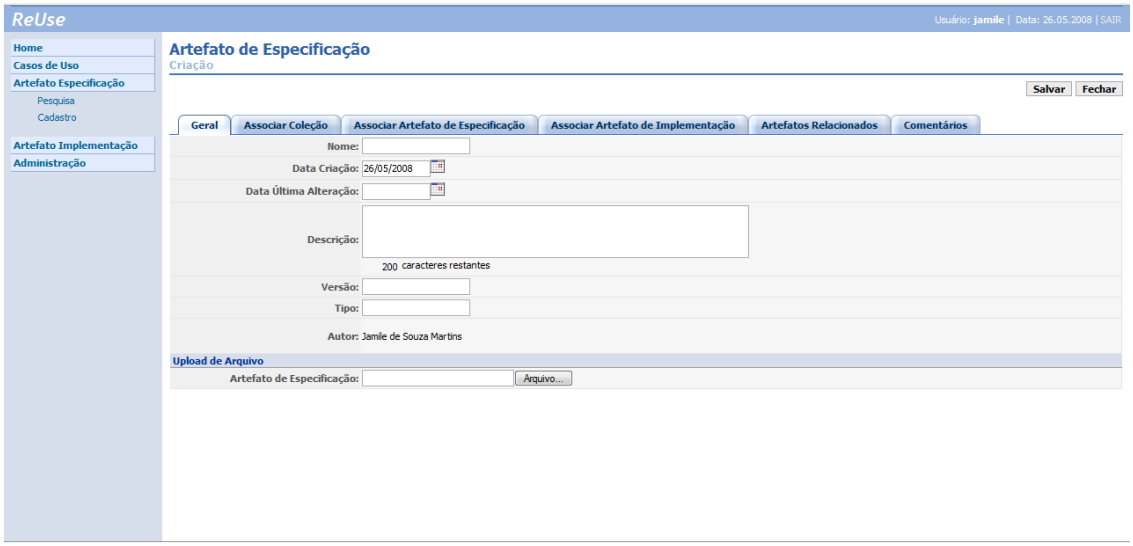

Figura 4.4: Tela de cadastro de artefato de especificação

Os filtros de pesquisa de artefatos de especificação são similares aos filtros de pesquisa de artefatos casos de uso. Os filtros de busca disponíveis são: identificador (corresponde ao identificador do artefato inserido na base de dados), nome, tipo, descrição, versão, coleção, data de criação, data da última alteração e autor.

Quaisquer dos filtros podem ser preenchidos de forma completa ou não. A ferramenta pesquisa as *substrings* informadas dentro dos campos cujos tipos são *string*. Um exemplo de resultado de pesquisa por um artefato de especificação pode ser visto na Figura 4.5

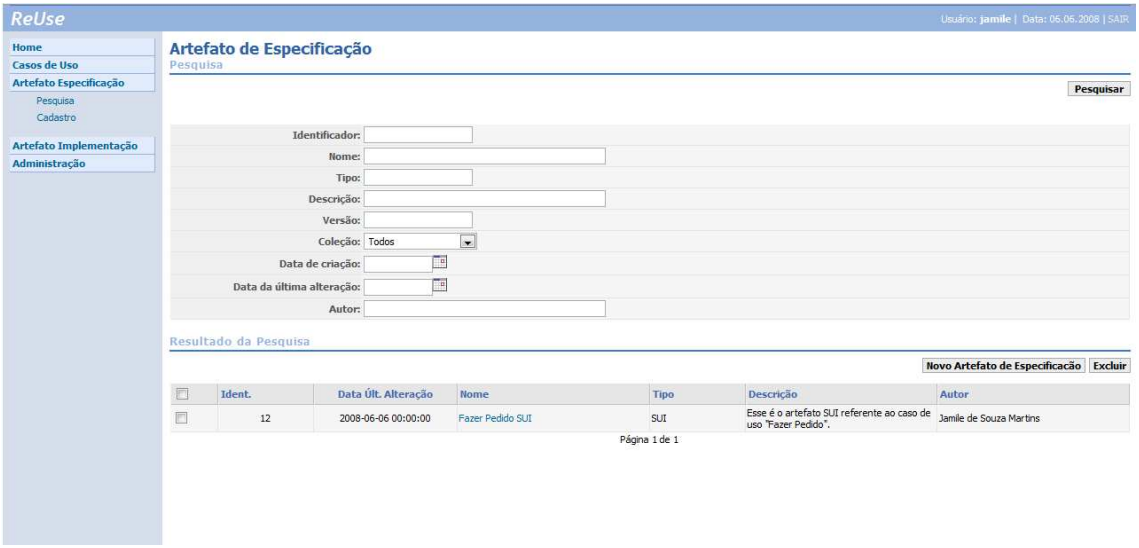

Figura 4.5: Tela de pesquisa de artefato de especificação

4.2.3 Artefato de Implementação

Os artefatos de implementação são os mais específicos da árvore de reificação. Correspondem às implementações das IUs descritas nos níveis anteriores (são as FUIs e as XUIs propostas no CRF).

O catálogo desse tipo de artefato é dividido em quatro pequenas etapas ou abas, enumeradas e descritas a seguir na ordem em que aparecem e pelo título correspondente:

1. **Aba Geral**: contém as informações essenciais referentes ao artefato de implementação. Essas informações são:

   - Nome: nome do artefato.
   - Data de criação: corresponde à data de criação do artefato ou à data em que ele foi inserido na base.
   - Data de última alteração: data em que o artefato sofreu a última modificação.
   - Descrição: é a descrição do artefato implementado.
   - Versão: corresponde à versão do artefato de implementação cadastrado no repositório.
   - Tipo: esse campo existe para que o usuário indique se o que está disponibilizando para *download* é um artefato executável (XUI) ou o código fonte de um artefato (FUI).
   - Linguagem de programação: indica a linguagem de programação em que o artefato foi implementado.
   - Autor: usuário que cadastrou o artefato na base de dados.

2. **Aba Associar Coleção**: similarmente aos artefatos casos de uso e aos artefatos de especificação, essa aba permite associar os artefatos de implementação a aplicações, *frameworks* ou qualquer outro elemento que interligue os artefatos de uma forma horizontal.

3. **Aba Artefatos Relacionados**: essa tela apresenta os artefatos de especificação relacionados com o artefato de implementação cadastrado.

4. **Aba Comentários**: os comentários dos artefatos de implementação têm o mesmo objetivo dos comentários de artefatos casos de uso e artefatos de especificação. Anotações podem ser feitas sobre os artefatos, possibilitando, assim, melhorias e refinamentos em versões posteriores.

Um exemplo de tela de inserção/edição de artefatos de implementação pode ser visto na Figura 4.6

Os filtros de busca disponíveis para artefatos de implementação são: identificador, nome, tipo, descrição, versão, linguagem de programação, coleção, data de criação, data da última alteração e autor.

Quaisquer dos filtros podem ser preenchidos de forma completa ou não. A ferramenta pesquisa as *substrings* informadas dentro dos campos cujos tipos são *string*. Um exemplo de resultado de pesquisa por um artefato de implementação pode ser visto na Figura 4.7.

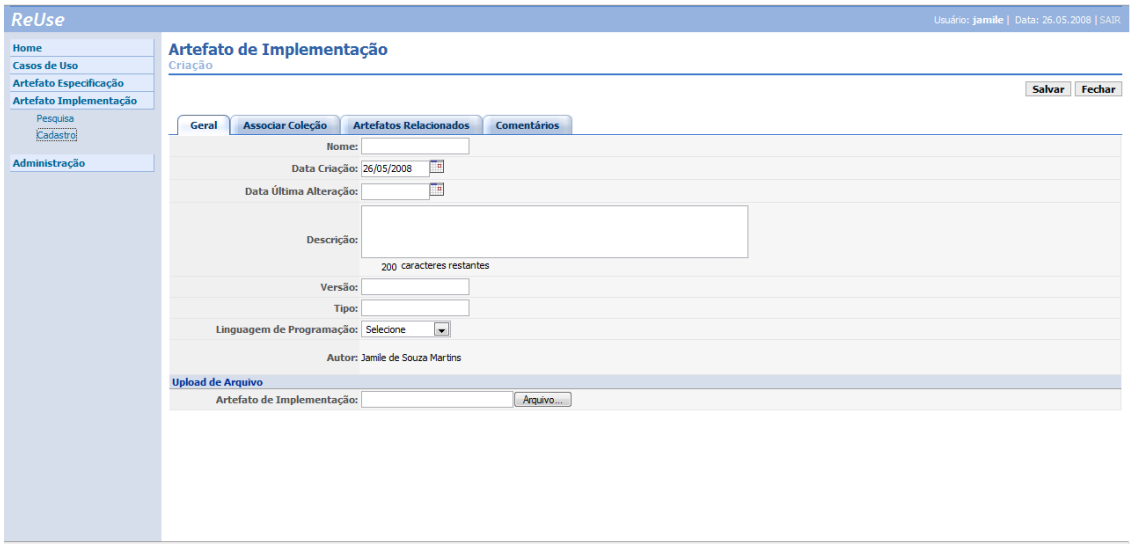

Figura 4.6: Tela de cadastro de artefato de implementação

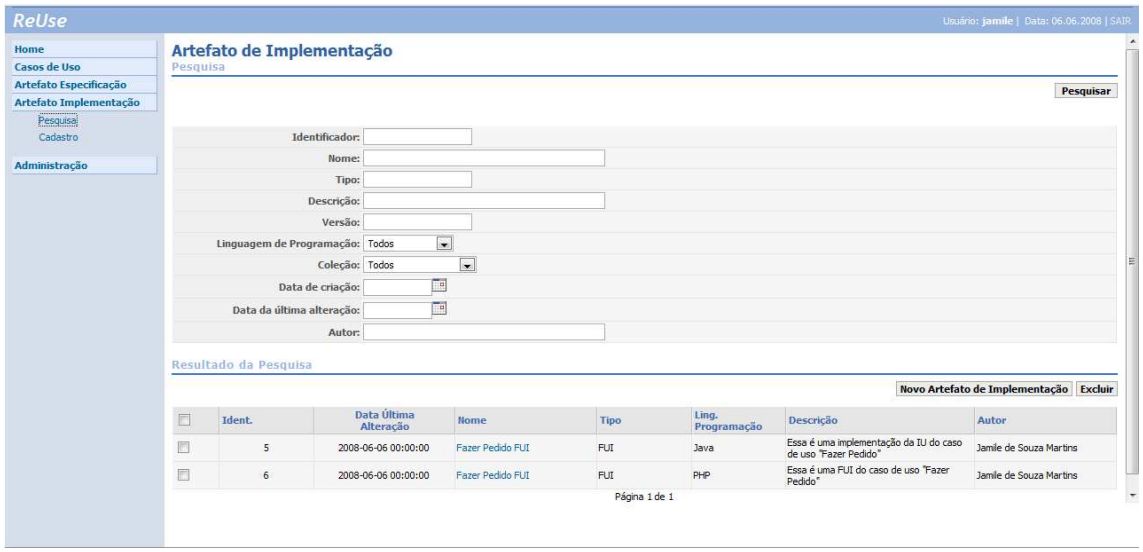

Figura 4.7: Tela de pesquisa de artefato de implementação

4.3 Funcionalidades de Suporte

Além dos módulos descritos acima, a ferramenta proposta contém, ainda, módulos auxiliares, para dar suporte às funcionalidades existentes. Os módulos auxiliares são: o módulo de cadastro de usuários, de cadastro de liguagens de programação, de cadastro de coleções (para agrupar os artefatos de uma forma horizontal), pacotes (para agrupar os casos de uso) e estatísticas.

Dentre os módulos auxiliares apresentados, o único deles que não apresenta a funcionalidade simples de cadastro é o módulo das estatísticas. Esse módulo foi criado para que se tenha informações dos artefatos com maior quantidade de *downloads* da ferramenta. Assim fica mais fácil para os administradores ou qualquer outro usuário do sistema colher informações sobre os artefatos mais reutilizados. Esse tipo de informação é de bastante utilidade quando se deseja mostrar os reais benefícios de utilização da ferramenta.

4.4 Exemplo de Caso de Uso Reificado Suportado Pela Ferramenta

Nessa seção, será apresentado um exemplo de caso de uso reificado a ser inserido na ferramenta ReUse. Adicionalmente, será apresentado um exemplo de pesquisa de um artefato, e como, a partir dele, pode-se chegar nos outro elementos que compõem sua árvore de reificação.

O caso de uso utilizado como exemplo para a inserção é apresentado na Tabela 4.1.

Caso de Uso: Fazer Pedido
Atores: Cliente
Pré-Condição: O usuário deve estar "logado" no sistema.
Fluxo de eventos primário
1. O caso de uso começa quando o cliente seleciona "fazer pedido".
2. O cliente fornece seu nome e endereço.
3. Se o cliente fornece apenas o CEP, o sistema preenche a cidade e o estado.
4. Enquanto o cliente quiser pedir itens faça
a.O cliente fornece código do produto
5. O sistema fornece a descrição e o preço do produto
6. O sistema atualiza o valor total
7. O cliente fornece as informações sobre cartão de crédito.
8. O cliente submete os dados ao sistema.
9. O sistema verifica as informações fornecidas, marca o pedido como "pendente" e envia as informações de pagamento para o sistema de contabilidade.
10. Quando o pagamento é confirmado, o pedido é marcado como "confirmado" e o número de pedido (NP) é dado ao cliente.
Fluxo de eventos secundário
A qualquer momento antes de submeter, o cliente pode selecionar cancelar. O pedido não é gravado e o caso de uso termina.
Pós-Condições: O pedido deve ter sido gravado e marcado como confirmado.

Tabela 4.1: Caso de Uso Fazer Pedido

Esse caso de uso faz parte de um sistema de compra e venda de produtos. Tal sistema possui duas versões: uma delas para ambiente *web* e a outra para ambiente *mobile* (para que a compra possa ser realizada através de um celular).

De acordo com o processo de definição de IU no PU, a partir da descrição do caso de uso, contrói-se um protótipo da IU. Esse protótipo pode ser tanto um rascunho feito em papel quanto um protótipo construído através de ferramentas. No caso do exemplo, o protótipo que descreve a IU do caso de uso da Tabela 4.1 é visto na Figura 4.8.

Considerando as implementações existentes para um ambiente *web* e para um ambiente *mobile*, juntamente com os artefatos descritos acima (caso de uso e protótipode IU), têm-se os elementos básicos para a construção/visualização da árvore de reificação.

Como elemento mais abstrato, o nodo raiz da árvore de reificação, tem-se o próprio caso de uso. Como artefato de especificação associado ao caso de uso, tem-se o rascunho da IU, que é mapeado para o elemento SUI da árvore de reificação. Finalmente, como artefatos de especificação, têm-se as implementações do caso de uso para as diferentes plataformas. Mesmo que a linguagem de programação utilizada seja a mesma, a arquitetura dos ambientes é diferente, logo, as implementações devem ser diferentes. Além disso, a versão para *web* pode ter sido implementada em uma linguagem e a versão para *mobile* em outra.

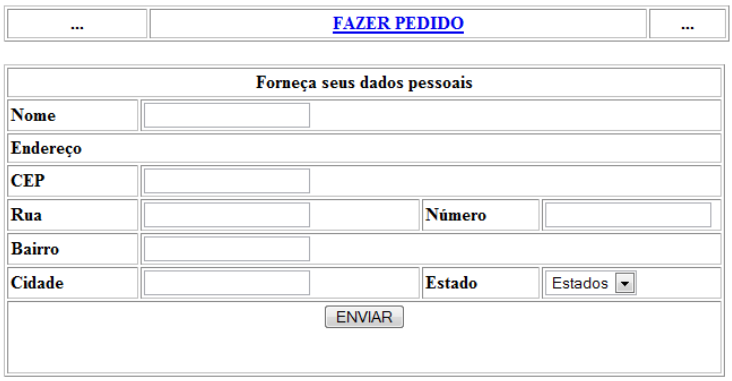

Figura 4.8: Protótipo da IU do caso de uso Fazer Pedido

De forma mais detalhada, poderiam ainda ter sido propostos os artefatos AUI e CUI do caso de uso acima. Um artefato AUI seria a descrição abstrata dos elementos que compõem a IU descrita pelo caso de uso. Em cima da AUI, seria construído um protótipo (ou vários), representando a SUI na árvore de reificação. Sobre a SUI, considerando o ambiente de interação e os elementos disponíveis para a construção da IU, seria construída uma (ou várias) CUI(s). Cada CUI teria suas implementações, disponibilizadas através de código-fonte (FUI) ou código executável (XUI) ou ambos.

Como exemplo de busca de um artefato, pode-se imaginar a tarefa de implementar em um sistema qualquer a autenticação (ou *login*) de um usuário. O responsável pela execução dessa tarefa (seja implementação ou análise ou mesmo ambas) tem em mente a visualização da tela de *login*, conhece a estrutura de árvore de reificação dos artefatos suportados pela ferramenta e sabe que um artefato de especificação SUI é um esboço de uma IU. Na tela de pesquisa de artefatos de especificação, mostrada na Figura 4.5, preenche-se o campo "Descrição" com a palavra "Login"e seleciona-se pesquisar. O resultado obtido, é mostrado na Figura 4.9.

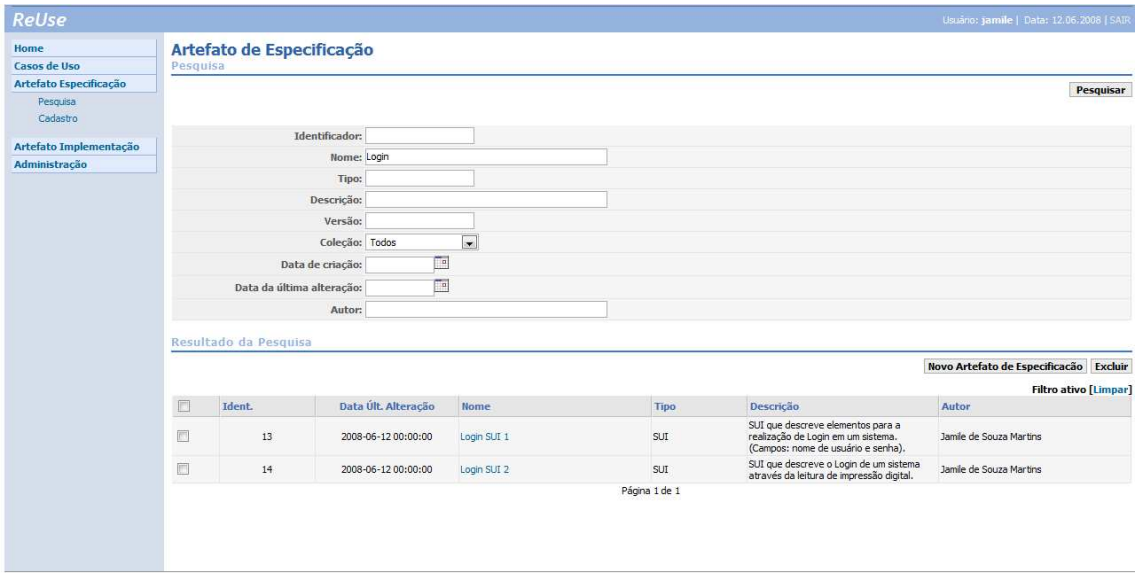

Figura 4.9: Resultado da pesquisa de artefatos de especificação Login

Dois artefatos de especificação, do tipo SUI, resultam dessa busca: um deles, sendo o rascunho de uma tela de *login* com campos "usuário" e "senha"e o outro, o rascunho

de uma tela de *login* cujo dado necessário para realizar a autenticação é uma impressão digital. O artefato escolhido é o de nome *Login SUI 1* (Figura 4.10), cujos campos a serem informados para o sistema são "usuário e "senha".

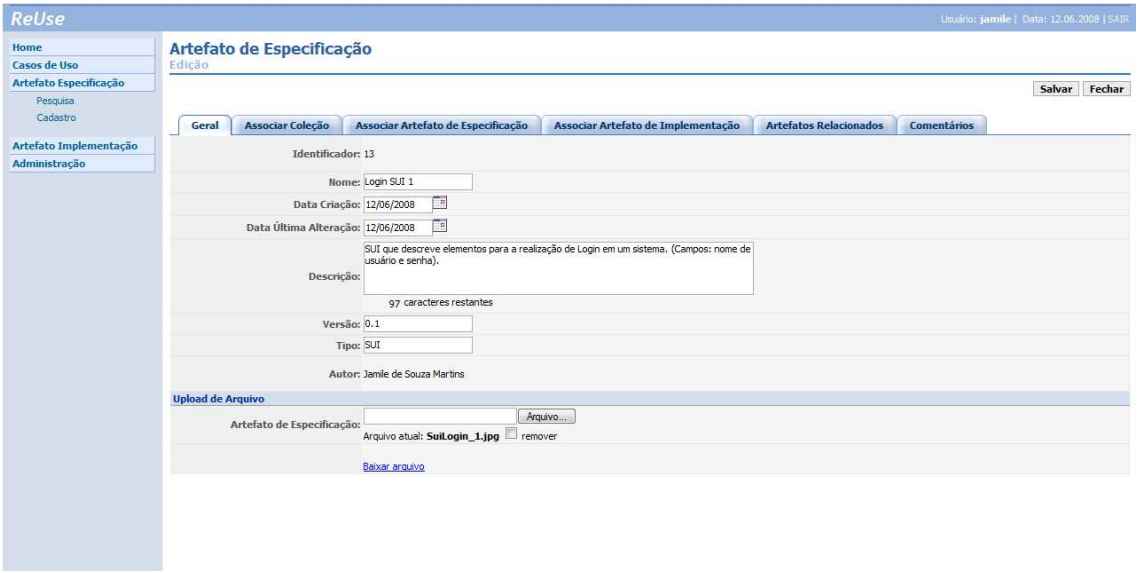

Figura 4.10: Artefato de especificação Login

No registro do artefato selecionado, encontra-se a SUI que o descreve, representada pela Figura 4.11.

Figura 4.11: SUI do artefato *Login SUI 1*

Uma vez encontrado um artefato de especificação (no caso do exemplo apresentado, a tela de *login*) que satisfaça a necessidade do usuário, pode-se verificar os demais artefatos associados a ele e a possibilidade de reusá-los. Estando no nível intermediário de artefatos (artefatos de especificação), pode-se procurar por artefatos de nível mais abstrato (caso de uso), de nível intermediário (artefatos de especificação) ou ainda de nível mais concreto (artefatos de implementação) associados ao artefatos encontrado.

A relação dos artefatos associados ao artefato de especificação encontrado, *Login SUI 1*, pode ser vista na Figura 4.12. Há dois artefatos associados ao artefato de especificação *Login SUI 1*: um deles é um caso de uso, que descreve a atividade de *login*; o outro é um artefato de implementação, uma FUI que implementa a tela de *login* descrita no artefato de especificação em linguagem PHP.

Figura 4.12: Artefatos associados ao artefato de especificação *Login SUI 1*

Assim, através da busca de um artefato de um nível intermediário, chega-se a artefatos mais abstratos e mais concretos, possibilitando o reuso nesses diversos níveis. Da mesma forma acontece na busca de casos de uso, onde pode se procurar por artefatos de especificação e através dos artefatos de especificação relacionados chegar aos artefatos de implementação e na busca por artefatos de implementação, onde se encontram os artefatos de especificação associados e através deles se chega aos casos de uso.

O cadastro e a pesquisa de todos os artefatos citados acima são suportados pela ferramenta proposta. O reuso desses artefatos se torna simplificado pois o caso de uso e a implementação foram divididos em artefatos menores, facilitando o processo de busca e reuso em diversas etapas do desenvolvimento. Nas atividades de modelagem e análise podem ser reusados casos de uso ou padrões de casos de uso e artefatos de especificação de um modo geral (AUIs, SUIs e CUIs), e nas atividades de desenvolvimento podem ser reusados os artefatos de implementação (FUIs e XUIs).

5 CONCLUSÕES E TRABALHOS FUTUROS

Os processos interativos de desenvolvimento de software, como visto nos capítulos anteriores, possuem diversas etapas. Em geral, começa-se com um artefato simples que vai sendo refinado ao longo do processo. A ferramenta apresentada neste trabalho oferece uma abordagem de busca por soluções existentes que pode ser inserida em diversas etapas do processo de desenvolvimento. Diferentemente de repositórios tradicionais, que oferecem apenas reuso de trechos de código fonte e/ou componentes, a ferramenta apresentada possibilita, devido aos diversos níveis de abstração de artefatos que ela suporta, que o reuso aconteça não somente no momento da codificação mas em etapas anteriores.

A estrutura da árvore de reificação e o exemplo apresentados no capítulo anterior ilustram de forma prática a construção de uma IU dividida em várias etapas e artefatos, que vão desde a especificação e modelagem da IU, através do caso de uso, até a sua implementação. Essa divisão dos elementos que compõem a hierarquia que constitui uma IU é o que possibilita o reuso em mais de uma etapa.

Tratando-se de questões mais práticas, referentes à ferramenta, pode-se considerar que as funcionalidades suportadas podem ser facilmente estendidas. Por ser uma ferramenta desenvolvida em uma liguagem bastante conhecida (PHP), e sobre um *framework* bastante intuitivo de se trabalhar, a adaptação da ferramenta para uma realidade mais próxima do usuário ou instituição que for utilizá-la fica mais fácil.

A ReUse é uma ferramenta de catalogação e busca de artefatos de desenvolvimento. Organizada sobre a estrutura da árvore de reificação, ela oferece diferentes formas de busca e catálogo para cada tipo de artefato. No entanto, como uma ferramenta que depende de entrada de dados não automatizada, ou seja, fornecida manualmente pelos usuários, a busca dos artefatos será tão eficiente quanto forem claros a descrição e os outros parâmetros fornecidos pelo usuário no momento da inserção do artefato na ferramenta.

Uma das principais dificuldades em se praticar reuso se refere à pesquisa de artefatos reusáveis. Segundo Krueger (KRU 92), o tempo do processo de busca e adaptação de um artefato já existente deve ser menor do que o tempo de construção de um artefato novo. Assim, um dos pontos de melhoria da ferramenta proposta, seria a construção de indexadores para a pesquisa. Um novo campo na pesquisa seria incluído para que o usuário informe exatamente o que procura (se um algoritmo, um componente, e assim por diante). Após a busca, o usuário informaria se encontrou o artefato que desejava. Em caso de sucesso, o valor desse campo extra da pesquisa seria incluído numa lista de termos do artefato encontrado. Assim, com o uso contínuo da ferramenta, teriam-se vários indexadores cadastrados e a busca se tornaria mais efetiva.

Outro trabalho futuro seria a ampliação do módulo de estatísticas, para conter informações que ajudassem a mensurar o quão efetiva é a ferramenta. Cada vez que um

usuário encontrasse por um artefato que necessita, ele preencheria uma enquente rápida informando se houve sucesso na busca, seu nível de satisfação ao utilizar a ferramenta, entre outras. Essas informações seriam catalogadas e auxiliariam a contabilizar, de forma prática, quão efetiva é a ferramenta proposta e quão corretamente as funcionalidades por ela providas estão sendo utilizadas. O sucesso da pesquisa por artefatos está diretamente relacionado à clareza dos dados informados no momento da inserção do artefato na ferramenta.

Enfim, enumerar as extensões possíveis de serem feitas numa ferramenta que pode ser tão complexa e completa quanto se queira é um trabalho sem fim. Como qualquer *software*, ele só pode ser aperfeiçoado se for utilizado. Assim, fica fácil descobrir os pontos de melhoria e as funcionalidades a serem incluídas que facilitariam o trabalho.

Por fim, é preciso reafirmar que boas ferramentas são importantes pontos de referência para se dar início à prática do reuso de uma forma mais efetiva. No entanto, há que se destacar o fator humano no processo de desenvolvimento e o quanto ele deve ser trabalhado para que uma prática, seja ela qual for, seja implantada num processo de desenvolvimento de software.

REFERÊNCIAS

[ALE 77] ALEXANDER, C. et al. **A Pattern Language**. New York: Oxford University Press, 1977.

[ALM 2007] ALMEIDA, E. et al. **Component Reuse in Software Engineering**. [S.l.]: R.I.S.E, 2007.

[AMB 2004] AMBLER, S. W. et al. **Extending the rup with the strategic reuse discipline**. Acessado em (2007). Disponível em: <http://www.enterpriseunifiedprocess.com /essays/strategicReuse.html>.

[BEZ 2002] BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. Rio de Janeiro: Elsevier, 2002.

[CAL 2003] CALVARY, G. et al. A Unifying Reference Framework for Multi-Target User Interfaces. **Interacting with Computers**, v.15, n.3, p.289–308, 2003.

[COC 2000] COCKBURN, A. **Writing Effective Use Cases**. [S.l.]: Addison-Wesley, 2000.

[COX 90] COX, B. J. Planning the Software Industrial Revolution. **IEEE Softw.**, Los Alamitos, CA, USA, v.7, n.6, p.25–33, 1990.

[EIS 2000] EISENSTEIN, J.; VANDERDONCKT, J.; PUERTA, A. Adapting to Mobile Contexts With User-Interface Modeling. In: THIRD IEEE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS (WMCSA'00), 2000, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2000. p.83.

[GAM 2000] GAMMA, E. et al. **Padrões de Projeto**: Soluções Reutilizáveis de Software Orientado a Objetos. [S.l.]: Bookman, 2000.

[JAC 92] JACOBSON, I. **Object-Oriented Software Engineering**: A Use Case Driven Approach. [S.l.]: Addison-Wesley, 1992.

[KOD 2007] KODERS. Acessado em (2007). Disponível em: <http://www.koders.com/>.

[KRU 2000] KRUCHTEN, P. **The Rational Unified Process - An Introduction, Second Edition**. [S.l.]: Addison-Wesley, 2000.

[KRU 92] KRUEGER, C. W. Software Reuse. **ACM Comput. Surv.**, New York, NY, USA, v.24, n.2, p.131–183, 1992.

[LAR 2001] LARMAN, C. **Applying Uml and Patterns**. [S.l.]: Addison-Wesley, 2001.

[LAR 2003] LARMAN, C. **Agile and Iterative Development**: A Manager's Guide. [S.l.]: Addison-Wesley, 2003.

[MCI 68] MCILROY. Mass produced software components. **Report on a conference by the NATO Science Committee**, p.138–150, 1968.

[MER 2007] MERO Base Component Finder. Acessado em (2007). Disponível em: <http://www.merobase.com>.

[MOR 2007] MOREIRA, A. A. **Reuso de IHC Orientado a Padrões, Dirigido por Casos de Uso e Integrado a um Processo de Desenvolvimento Baseado em UML**. 2007. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.

[MYS 2007] MYSQL. Acessado em (2007). Disponível em: <http://www.mysql.com/>.

[OMG 2005] OMG. **OMG - Object Management Group, Reusable Asset Specification**. Acessado em (2007). Disponível em: <http://www.omg.org/docs/formal/05-11-02.pdf>.

[OMG 2007] OMG. **OMG - Object Management Group, Unified Modeling Language Specification Version**. Acessado em (2007). Disponível em: <http://www.omg.org/docs/formal/07-11-04.pdf>.

[PHP 2007] PHP Hypertext Preprocessor. Acessado em (2007). Disponível em: <http://www.php.net/>.

[PIM 2000] PIMENTA, M. S. TAREFA: Uma Abordagem para Engenharia de Requisitos de Sistemas Interativos. **Proceedings of Jornadas Iberoamericanas de Ingeniería de Requisitos y Ambientes Software**, 2000.

[REZ 2007] REZENDE, M. **phpwafr**. Versão 1.1.2. Disponível em: <http://sourceforge.net/projects/phpwafr>.

[TRA 88] TRACZ, W. Software Reuse Myths. **SIGSOFT Softw. Eng. Notes**, New York, NY, USA, v.13, n.1, p.17–21, 1988.

[UFR 2007] UFRJ, C. **Odissey**. Acessado em (2007). Disponível em: <http://reuse.cos.ufrj.br>.

[VAN 93] VANDERDONCKT, J. M.; BODART, F. Encapsulating knowledge for intelligent automatic interaction objects selection. In: CHI '93: PROCEEDINGS OF THE INTERACT '93 AND CHI '93 CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 1993, New York, NY, USA. **Anais...** ACM, 1993. p.424–429.

[YON 98] YONGBEON K. STOHR, E. Software Reuse: Survey and Research Directions. **Journal of Management Information Systems**, v.14, n.4, p.113–147, 1998.

[ZIR 95] ZIRBES, S. F. **A Reutilização de Modelos de Requisitos de Sistemas por Analogia**: Experimentação e Conclusões. 1995. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.