Uma Proposta de um Repositório de Padrões de Software Integrado ao RUP[1]

Fabiana Marinho[*], Misael Santos, Rute Nogueira Pinto e Rossana Andrade
[*] Instituto Atlântico, Rua Chico Lemos, 94, 660822-780, Fortaleza - Ceará – Brasil,
Universidade Federal do Ceará, Departamento de Computação, Campus do Pici
Bloco 910, 60455-760, Fortaleza - Ceará - Brasil
fabiana@atlantico.com.br,{misael,rute,rossana}@lia.ufc.br

Resumo

Com o crescimento tanto do número de padrões de software para análise, projeto e implementação quanto da reutilização dos mesmos no desenvolvimento de sistemas, surge a necessidade de facilitar a busca e aplicação desses padrões. Esse artigo visa apresentar uma proposta de integração de padrões de software armazenados em um repositório com um modelo de processo de desenvolvimento. O Rational Unified Process (RUP), tem sido utilizado comercialmente com sucesso e é o modelo escolhido para esta proposta.

Abstract

With the increase number not only of software patterns for analysis, design, project and implementation but also of pattern reuse in the system development process, there is a need for finding an appropriate mechanism for searching and applying these patterns. This work aims to present a proposal to integrate software patterns available in a repository with a development process model. Rational Unified Process (RUP) has been successfully applied commercially and it is the chosen process for this proposal.

1. Introdução

Uma grande quantidade de padrões de software de análise, de projeto e de implementação (denominados idiomas) estão disponíveis em Web sites e na literatura [4][7][12][13][20], prontos para serem reutilizados por engenheiros de software nas diferentes fases de desenvolvimento de software. Embora uma tentativa de catalogar os padrões existentes e já publicados em conferências *Pattern Languages of Programming* (PLoP) tenha sido apresentada em [23], os padrões são divididos por categoria e fica a critério do engenheiro de software fazer uma busca exaustiva para decidir o padrão correto para resolver determinado problema.

Já existem ferramentas que se propõem a facilitar a aplicação de padrões de projeto no desenvolvimento de software, auxiliando na modelagem de classes e incluindo a geração automática de código [2][3][6][8][14][15][19][27]. Apesar disso, é difícil encontrar um repositório que contenha padrões aplicáveis a diferentes fases do processo de desenvolvimento e que auxilie o engenheiro de software na busca e reutilização desses padrões de forma integrada ao processo.

Em [25], um processo de desenvolvimento baseado na reutilização de componentes nas fases de requisitos e projeto da arquitetura é proposto. Neste processo, a reutilização de um componente é limitada pelas decisões de projeto detalhadas na implementação desse

[1] O Instituto Atlântico e o CNPq suportam financeiramente este trabalho

componente. Essas decisões podem causar conflitos com as necessidades do usuário, tornando a reutilização do componente impossível ou introduzindo ineficiências significantes na aplicação que está sendo desenvolvida. A reutilização de padrões não possui este tipo de restrição. Os padrões de software são mais abstratos, podendo ser implementados de forma a se adaptar às necessidades específicas da aplicação que está sendo desenvolvida.

O objetivo deste trabalho é apresentar uma proposta de um repositório de padrões de software e a sua integração com um modelo de processo de desenvolvimento de software. O Rational Unified Process (RUP) é o processo de desenvolvimento de software escolhido. A integração proposta pode ser generalizada para qualquer modelo de processo de desenvolvimento que tenha as características de incremental e iterativo. O RUP foi escolhido por possuir as notações mais apropriadas para a reutilização de padrões nas diferentes fases de desenvolvimento, além de já estar sendo largamente adotado por inúmeras empresas nacionais.

A proposta deste artigo não pretende modificar o RUP original e nem substituir o suporte a padrões de projeto que o mesmo possui, mas oferecer aos engenheiros de software um conjunto mais abrangente de padrões de requisitos, análise e projeto que possa ser consultado e adaptado durante as fases de desenvolvimento. Para isso, é apresentado o protótipo de um repositório de padrões que será usado para catalogar os padrões que servirão de insumo para diferentes fases do RUP. A integração do padrão selecionado com o sistema que está sendo desenvolvido não vai acontecer automaticamente. O engenheiro de software será responsável por interpretar e adaptar o padrão de acordo com as suas necessidades.

Este trabalho está organizado da seguinte forma, na seção 2 apresentamos uma breve descrição dos trabalhos relacionados à nossa proposta. Na seção 3 analisamos a reutilização de padrões de software em sistemas comerciais a fim de enfatizar a motivação deste trabalho. O protótipo do repositório de padrões é descrito na seção 4. Na seção 5 descrevemos o ciclo de vida adotado pelo RUP e a nossa proposta para a integração do repositório de padrões com o mesmo. Finalmente, a Seção 6 contém nossas conclusões e direcionamentos para trabalhos futuros.

2.Trabalhos Relacionados

Um catálogo de padrões de software [23] e uma proposta de reutilização de componentes [25] foram brevemente discutidos na Seção 1. Ferramentas para o auxílio à aplicação de padrões de software que trabalham com geração automática de código e de modelagem de classes também estão disponíveis na literatura [3][6][8][19][27] e são introduzidas a seguir.

Budinsky et al apresentam em [3] uma das primeiras ferramentas para a geração automática de código para padrões. A ferramenta adota o formato apresentado em [13] para o template de padrões e gera códigos referentes aos padrões a partir de alguns parâmetros informados pelo usuário. O processo de *wizard* permite que a implementação gerada do padrão esteja adequada às necessidades do problema.

O UMLStudio [27] é uma ferramenta comercial de modelagem de classes em UML que possui um catálogo de padrões que podem ser selecionados em uma lista e inseridos na modelagem. A ferramenta inclui a modelagem dos seguintes padrões [13]: Command, Iterator, Memento, Abstract Factory, Factory Method, Adapter e Proxy. A inserção é feita de forma estática, ou seja, qualquer interação com classes já existentes, ou modificações nas classes ou nos métodos devem ser feitas manualmente após a inserção.

Uma abordagem mais dinâmica para a automação de padrões é apresenta pelo

ModelMaker [19] e pelo Together [2].

O ModelMaker, é uma ferramenta comercial com suporte à modelagem de dados em UML, geração de classes e geração de pacotes de componentes para o Borland Delphi. O ModelMaker implementa os seguintes padrões de [13]: Adapter, Mediator, Singleton, Decorator, Visitor e Observer. Além disso, ela implementa os padrões *Lock* e *Reference Count*. A ferramenta tem um nível de automação avançado, por exemplo, classes envolvidas com padrões respondem de maneira automática e inteligente às mudanças feitas pelo projetista em outras partes do modelo UML.

O CodePro Studio [6] é uma ferramenta comercial que também é integrado a ambientes Java de desenvolvimento como o Eclipse [11] e o Websphere Studio [14]. O módulo Java Pattern Wizard da ferramenta permite gerar classes implementando vários padrões de projeto e outros padrões de implementação gerais para Java. Os padrões catalogados na ferramenta são divididos em 6 categorias, a seguir: padrões de Criação, Comportamentais, Estruturais, GUI (Interface Gráfica), J2EE e Teste. Novos padrões podem ser inseridos ao repositório criando novas instâncias xml dos elementos que definem a estrutura padrões. A estrutura dos padrões definida pelo sistema é bastante limitada (id, name, icon, description, category, source e strategy), o que faz com que o usuário não tenha acesso ao padrão no seu formato original.

Outra exemplo de ferramentas que trabalham com padrões de software são as ferramentas de *Refactoring* [10][15][16], que utilizam técnicas de reestruturação de código visando aumentar a legibilidade e manutenibilidade do sistema, sem contudo alterar o seu comportamento. O *Refactoring* é realizado através da incorporação de um novo padrão ao código.

Em [8] é apresentado o protótipo de um ambiente de desenvolvimento para a definição e reutilização de padrões. No trabalho é criado um formalismo, denominado P-Sigma, para a representação de padrões e de sistemas de padrões. Além de representar os elementos de um *template* de padrões (identificador, classificação, problema, contexto, forças, solução, diagramas e conseqüências), os relacionamentos entre os padrões (usa, refina, requer e alternativo) são representados.

Vale a pena mencionar nesta seção o trabalho desenvolvido em [17], onde os autores introduzem uma ferramenta de auxílio ao desenvolvimento de sistemas baseado em componentes. O objetivo do trabalho é propiciar técnicas seguras para desenvolvimento de sistemas em grupos de trabalho, com políticas de manutenção e ferramentas de auxílio à reutilização de software. O repositório de componentes usado, além das informações técnicas dos componentes (endereço do arquivo, tamanho, criador, data de inclusão e atualização, etc.), armazena características dos componentes, tais como Contexto, Problema e Solução que compõem parte de um *template* dos padrões de software. O trabalho possui um mecanismo de busca semelhante ao proposto pelo nosso artigo. Existem dois tipos de busca, uma simples que utiliza a pesquisa entre as palavras-chave inseridas no registro de cada componente, e uma busca avançada aonde a pesquisa é realizada nas características do componente.

Em [5], as deficiências apresentadas pelas ferramentas de geração de código foram discutidas. Por exemplo, a forma de integração do código gerado ao código existente pode ser utilizado para ajudar a diferenciar as ferramentas analisadas.

Uma das diferenças entre o trabalho proposto neste artigo e os demais trabalhos referidos acima é que no nosso trabalho existe uma preocupação em dar suporte à inserção de vários tipos de padrões, não restrito a um número limitado de padrões de projeto como em [19][27]. Além disso, o engenheiro de software visualizará o padrão no seu *template* original, ficando a seu critério adaptar o padrão às suas necessidades.

Outra diferença importante é quanto à classificação dos padrões. Usamos a classificação baseada nas categorias citadas em [23], que consiste em classificar os padrões tanto quanto às fases de desenvolvimento nas quais eles podem ser aplicados (i.e., análise, projeto, implementação), quanto à natureza do padrão (e.g., comportamental, estrutural e arquitetônico), quanto ao domínio de aplicação específico (e.g., Web, interface, segurança e sistemas distribuídos), garantindo assim uma classificação mais abrangente e tornando também mais eficiente o mecanismo de busca no repositório e a aplicação dos padrões durante as diferentes fases do processo de desenvolvimento.

3. Considerações sobre a Reutilização de Padrões

Nesta seção, é analisado o impacto da reutilização de padrões dentro do processo de desenvolvimento de software no Instituto Atlântico (IA), que é uma instituição de pesquisa e desenvolvimento localizada em Fortaleza, Ceará. O IA vem desenvolvendo software em diversas áreas tecnológicas, posicionando-se como fonte inovadora de conhecimento e de geração de resultados tecnológicos. Aplicações de software para sistemas de suporte a negócios e operações, para engenharia, planejamento, inteligência de negócios, para serviços voltados à Internet e diversos outros setores de telecomunicações, de energia e do governo fazem parte do escopo de atuação do IA. O RUP é utilizado pelas equipes do IA para o processo de desenvolvimento de software.

Para identificar que padrões de software estavam sendo aplicados foram realizadas entrevistas com dois integrantes de cada um dos seis projetos (cada sistema a ser desenvolvido no IA é associado a um projeto) nas fases finais do processo de desenvolvimento. Levando-se em consideração as entrevistas, foi identificada a aplicação de padrões em todos os projetos investigados, embora em fases distintas do processo de desenvolvimento. Verificou-se, principalmente, a aplicação dos padrões de projeto da *Gang of Four* (GoF) [13], os padrões orientados a arquitetura de software (POSA) [4] e os padrões do catálogo *Core J2EE* [9].

De acordo com os resultados das entrevistas, as equipes foram estimuladas a utilizar padrões ou por membros do mesmo projeto ou de outros projetos do IA ou ainda pelos próprios clientes. A existência de um *framework* do próprio Instituto Atlântico contendo padrões de projeto já implementados funcionou como um estímulo para a aplicação de padrões.

As principais vantagens citadas durante as entrevistas sobre a reutilização de padrões estão descritas abaixo:

- evolução de código;
- modularidade;
- desacoplamento entre áreas de responsabilidades de forma que as mudanças em uma não ocasionem mudanças nas outras;
- diminuição da complexidade do projeto e do código final;
- estabilidade do código;
- os padrões em uso são genéricos o suficiente para se encaixarem bem em projetos distintos;
- confiabilidade na reutilização de padrões cujas soluções são comprovadas;
- ganho de produtividade;
- facilidade de repassar conhecimento entre os engenheiros de software experientes; e
- facilidade de aprendizado de novas áreas de conhecimento para a equipe sem experiência na aplicação a ser desenvolvida.

Esse relato do impacto positivo indicou que a reutilização de padrões funciona como um instrumento eficaz para aumentar a produtividade, diminuir o custo de desenvolvimento e promover a integração e comunicação entre projetos. Daí a importância de estimular a busca e a reutilização de padrões durante as diferentes fases de desenvolvimento de software.

Por outro lado, quando padrões são aplicados nos projetos sem um entendimento prévio da solução proposta, das conseqüências de sua aplicação e do relacionamento com outros padrões, um impacto negativo que pode ser observado é a complexidade de código com o aumento do número de classes (no caso de reutilização de padrões de projeto).

É importante ainda conservar entre os membros da equipe de desenvolvimento um senso crítico de que padrões não são soluções para todos os problemas durante o desenvolvimento de software e de que nem sempre a reutilização de padrões acarreta uma diminuição da complexidade do código final e um aumento da qualidade do produto final. Entretanto, a reutilização de padrões é uma boa prática da engenharia de software e um dos caminhos para atingir a qualidade.

A experiência relatada pelos integrantes dos projetos analisados consolidou a recomendação da comunidade de padrões de que a opção pela utilização de padrões deve ser feita nas fases iniciais de desenvolvimento para ressaltar as vantagens e evitar as desvantagens citadas anteriormente. Com a integração de um procedimento de busca em um repositório de padrões e a aplicação dos padrões recuperados nas fases iniciais de desenvolvimento de sistemas utilizando o RUP, este trabalho pretende contribuir para agilizar a reutilização de um conjunto mais extenso de padrões disponíveis na Web e na literatura. Além disso, este trabalho pretende contribuir com as eventuais discussões entre as equipes de desenvolvimento quanto à fase onde cada padrão deve ser aplicado. O IA é parceiro no desenvolvimento deste trabalho.

4. O Repositório de Padrões

No repositório, os padrões que servem de insumo para as fases do processo de desenvolvimento do RUP são armazenados. Esta seção descreve as funcionalidades que serão implementadas no protótipo do repositório de padrões e os diagramas de caso de uso para os dois atores envolvidos no repositório: o usuário e o administrador. Além disso, os mecanismos de busca de padrões utilizados são apresentados.

4.1. Definição do Repositório

As funcionalidades que serão implementadas no protótipo do repositório de padrões são descritas a seguir:

- Cadastrar linguagem de padrões: o repositório deve permitir o cadastro de linguagens de padrões. Padrões pertencentes a uma mesma linguagem devem estar relacionados de acordo com [9] que descreve uma relação entre contexto e contexto resultante em padrões pertencentes a uma linguagem de padrões. Para este caso, a partir do formalismo de [8], o seguinte relacionamento foi criado:
    - Se os padrões P1 e P2 pertencem a uma mesma linguagem de padrões L1 e possuem o relacionamento de dependência nesta linguagem P2 *depende* de P1, então o contexto do padrão P2 deve ser expresso usando o contexto resultante de P1.
- Cadastrar categoria de padrões: o repositório deve permitir a classificação dos padrões com o cadastro de categorias dos padrões. As principais categorias consideradas estão

baseadas em [4][12][13]: padrões de análise, padrões de projeto, padrões estruturais, padrões comportamentais, entre outros. Categorias específicas para o domínio das aplicações como sugerida em [23] também são permitidas, por exemplo: padrões de segurança, de interface e Web.

- Cadastrar sistema: o repositório deve realizar o cadastro de sistemas que utilizaram padrões no seu desenvolvimento. Dessa forma, será permitido ao usuário verificar posteriormente em quais sistemas um determinado padrão foi aplicado, adicionalmente aos usos conhecidos apresentados no próprio padrão.
- Cadastrar padrão: o repositório deve permitir a inclusão de padrões em uma base de dados, adotando um *template* genérico que seja capaz de englobar os vários formatos de padrões existente na literatura, por exemplo [4][9][13][18]. O *template* adotado consiste, inicialmente, de: Nome, Analogia, Contexto, Problema, Intenção, Aplicabilidade, Solução, Forças, Resumo, Implementação, Estrutura, Contexto Resultante, Conseqüências, Padrões Relacionados, Exemplos, Sintomas, Forças Contrárias, Usos Conhecidos e Referências. A intenção é dar suporte aos mais variados tipos de padrões de software existentes, mantendo o seu formato original. Esse formato semi-estruturado dos dados incentivou o uso de XML para armazenamento. Durante a inclusão dos padrões relacionados, o administrador pode definir relacionamentos de acordo com [8], que apresenta os seguintes relacionamentos possíveis entre padrões:
    - Se um padrão P_1 *usa* um padrão P_2, então a solução do padrão P_1 deve ser expressa usando P_2.
    - Se um padrão P_1 *refina* um padrão P_2, então o problema do padrão P_1 deve ser uma especialização do padrão P_2.
    - Se um padrão P_1 *requer* um padrão P_2, então aplicação do padrão P_2 é exigida na aplicação de P_1.
    - Se um padrão P1 é *alternativo* a um padrão P2, então os padrões P1 e P2 fornecem soluções diferentes para o mesmo problema.
- Modificar padrão: o repositório deve fornecer um mecanismo de atualização dos dados de um padrão cadastrado.
- Excluir padrão: o repositório deve fornecer um mecanismo de exclusão de um padrão cadastrado. Antes de excluir o padrão, a ferramenta deve verificar se existem padrões que referenciam o padrão a ser excluído. Duas opções devem ser oferecidas: exclusão das referências ou manutenção das mesmas. Em caso de manutenção, o link ao conteúdo do padrão deve ser desfeito, mas o nome do padrão permanece no item padrões relacionados.
- Realizar busca: o objetivo principal do repositório é tornar os padrões disponíveis para busca e recuperação. Alguns itens devem ser adicionados ao banco de dados para facilitar os mecanismos de busca que serão implementados no repositório de dados (veja maiores detalhes na seção 4.2.). Após a pesquisa, todos os dados e arquivos relacionados ao(s) padrão(ões) encontrados devem estar disponíveis ao usuário. Outra importante informação resultante dessa busca é a visualização de sistemas desenvolvidos que já aplicaram aquele padrão.

O administrador do sistema e o usuário do sistema são os dois atores identificados para o protótipo do repositório de padrões (veja Figura 1). O principal objetivo do usuário do repositório é reutilizar um ou mais padrões. Portanto, a funcionalidade mais importante para o usuário é a captura dos padrões dentro do repositório de dados que possam ser usados no desenvolvimento de sua aplicação (busca de padrões). O administrador deve ser capaz de

manter os padrões (cadastro, alteração e exclusão). As principais atividades do administrador e do usuário do repositório de padrões podem ser sumarizadas nos diagramas de casos de uso mostrado na Figura 1.

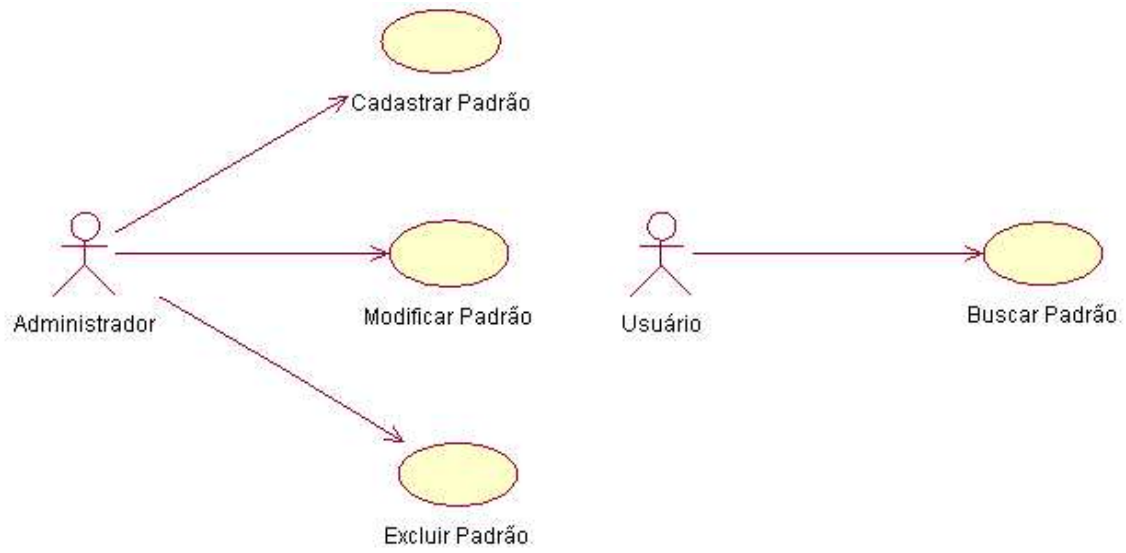

Figura 1 – Diagrama de Casos do Protótipo do Repositório de Padrões

4.2. Mecanismos de Busca

O mecanismo de busca do repositório consiste em uma busca por palavras chave entre campos selecionados do *template* dos padrões. O formalismo para a entrada da pesquisa é semelhante à maioria dos sistemas de busca da Web conhecidas como Search Engines [26]. Operadores lógicos, tais como “e” e “ou”, também devem ser aceitos. O texto de consulta passa então por um processamento para que sejam definidos todos os parâmetros da consulta, palavras e operadores, que serão acompanhados dos campos de consulta selecionados.

O usuário pode realizar buscas sobre todos os padrões catalogados no repositório de padrões. Os padrões que correspondem à busca realizada são capturados e visualizados. Dois mecanismos de busca para o repositório de padrões são adotados: busca simples e busca avançada. Esses mecanismos são descritos a seguir.

- **Busca Simples**

A busca simples consiste inicialmente da pesquisa nas palavras chave do padrão a partir do texto de entrada. A Figura 2 mostra a janela do protótipo do repositório de padrões que será utilizada para a realização de busca simples. No frame esquerdo, estão localizados os parâmetros da consulta e um link para o modo de Busca Avançada. No frame direito, são listados os resultados encontrados.

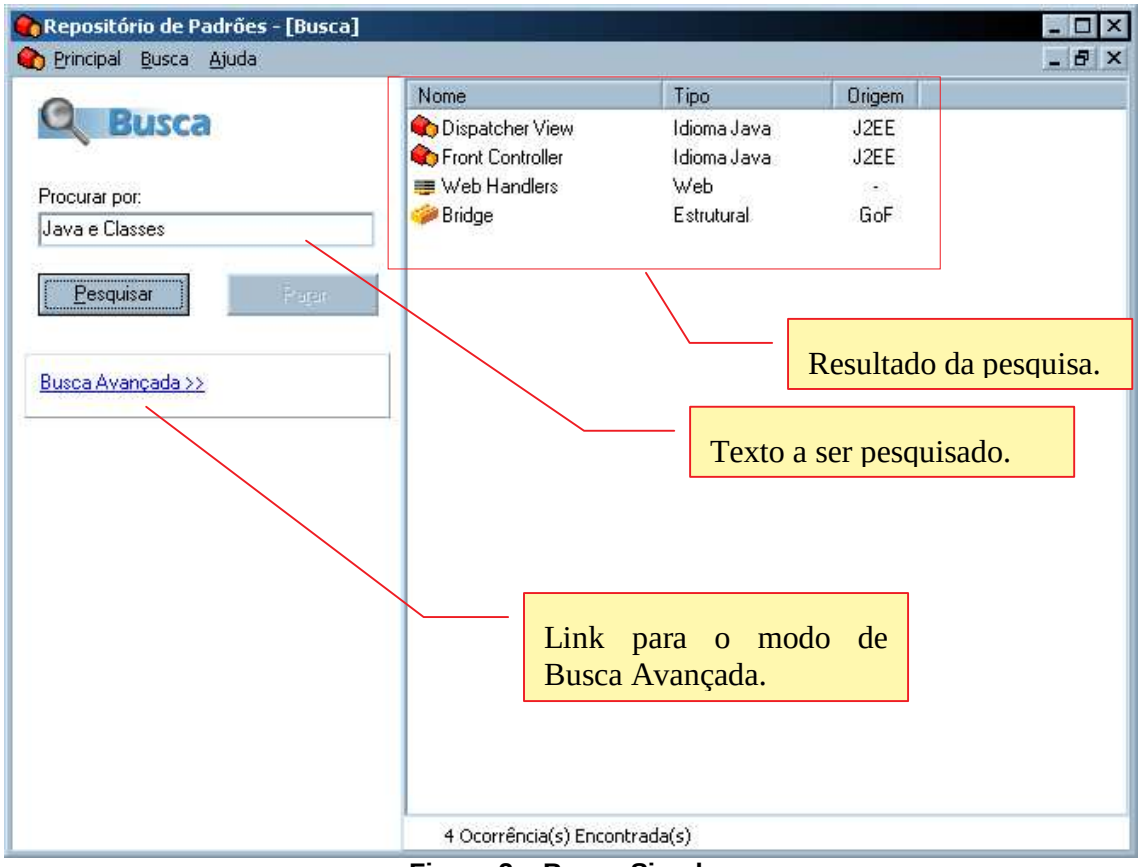

Figura 2 – Busca Simples

- **Busca Avançada**

A busca avançada permite que sejam selecionados múltiplos campos para a consulta. Pode-se ainda refinar a pesquisa selecionando a categoria do padrão procurado e/ou um sistema, a fim de buscar somente os padrões utilizados no seu desenvolvimento. A Figura 3 mostra a janela do protótipo do repositório de padrões que será utilizada para a realização de busca avançada. No frame esquerdo, ficam localizados os parâmetros da consulta que consistem no campo texto de consulta, nas opções de seleção múltipla de campos de pesquisa, na seleção da categoria do padrão, seleção de sistema e em um link para o modo de Busca Simples. Assim como na Busca Simples os resultados encontrados são listados no frame direito da janela.

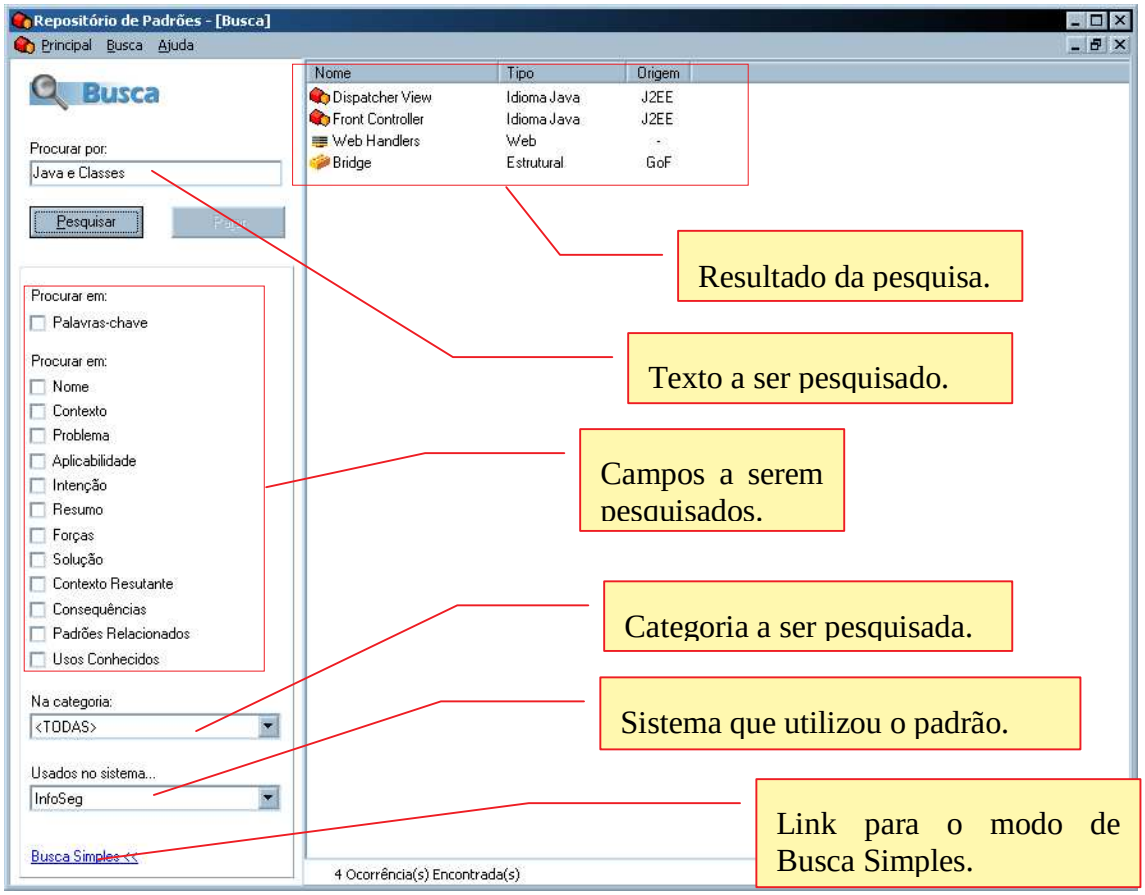

Figura 3 – Busca Avançada

5. Integração do Repositório de Padrões com o RUP

A aplicação de padrões pode ocorrer em diferentes fases do processo de desenvolvimento de software. Uma ferramenta que se proponha a armazenar os padrões existentes na literatura e que permita buscar aqueles padrões que sejam mais adequados para a solução de um determinado problema, pode facilitar bastante essa tarefa. Nesta seção, apresentamos como o repositório de padrões pode ser é integrado ao modelo de processo de desenvolvimento de software do RUP.

5.1. Rational Unified Process - RUP

O RUP é um processo iterativo e incremental que provê uma abordagem disciplinada para o desenvolvimento de software [21]. Na engenharia de software, um modelo de processo é um conjunto de passos parcialmente ordenados com a intenção de construir um produto de software de qualidade, capaz de atender às necessidades e exigências do usuário final de acordo com planejamento e orçamento previstos [25].

O RUP encoraja o controle de qualidade e o gerenciamento de riscos contínuos e objetivos. A avaliação da qualidade é inserida no processo em todas as atividades. O gerenciamento de riscos é inserido no processo de forma que os riscos que se opõem ao

sucesso do projeto sejam identificados e atacados no início do processo de desenvolvimento. Além disso, o desenvolvimento é centrado na arquitetura. Uma arquitetura robusta minimiza o re-trabalho e aumenta a reutilização de componentes e a capacidade de manutenção do sistema.

Conforme apresentado na Figura 4, o RUP possui duas dimensões. O eixo horizontal representa o aspecto dinâmico do processo e mostra aspectos do ciclo de vida à medida que este se desenvolve. O eixo vertical representa o aspecto estático do processo, como ele é descrito em termos de componentes, disciplinas, atividades, fluxos de trabalho, artefatos e papéis do processo [22].

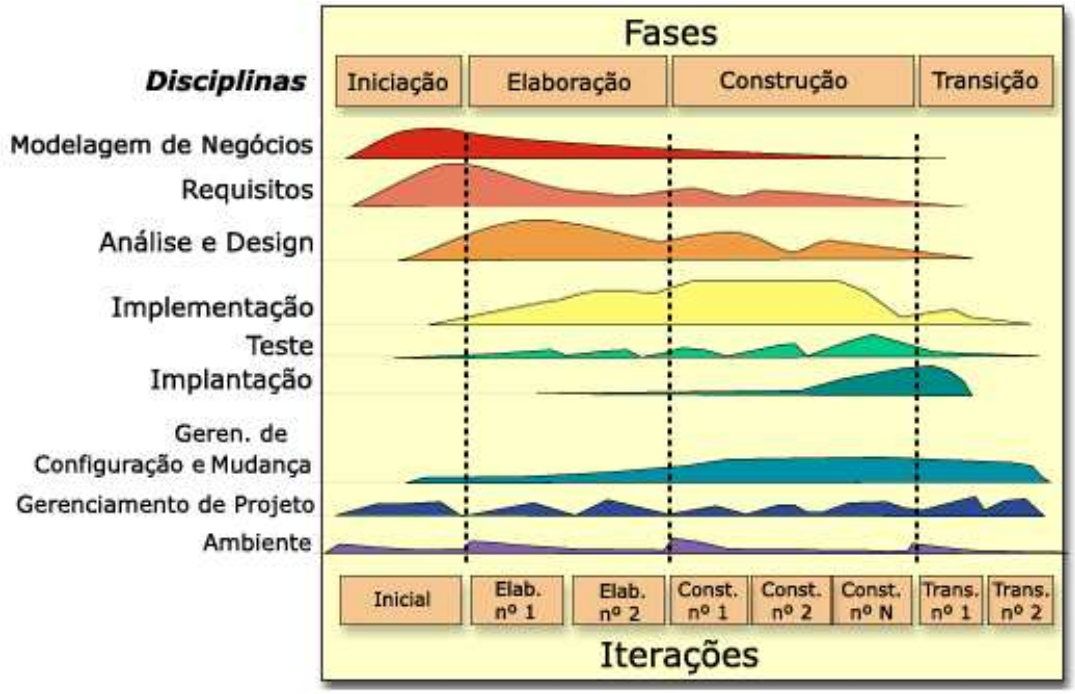

Figura 4 – Ciclo de vida de desenvolvimento do RUP[1]

O ciclo de vida de software do RUP é dividido em quatro fases seqüenciais: iniciação (termo adotado na tradução oficial da *Rational* para Português), elaboração, construção e transição [22].

O objetivo da fase de iniciação é delimitar o escopo do projeto, discriminando os principais cenários de operação. Além disso, custos, planos e riscos são estimados. A fase de elaboração tem por objetivo analisar o domínio do problema, de modo a propor uma arquitetura para os cenários da aplicação. O objetivo da fase de construção é esclarecer os requisitos restantes e concluir o desenvolvimento do sistema com base na arquitetura definida. Finalmente, a fase de transição tem por objetivo assegurar que o software esteja disponível para seus usuários finais. A fase de transição inclui testar o produto e realizar pequenos ajustes com base no feedback do usuário. No fim do ciclo de vida da fase de transição, os objetivos devem ter sido atendidos e o projeto deve estar em uma posição para fechamento. Em alguns casos, o fim do ciclo de vida atual pode coincidir com o início de outro ciclo de vida no mesmo produto, conduzindo à nova geração ou versão do produto. Para outros projetos, o fim da fase de transição pode coincidir com uma liberação total do produto a terceiros que poderão ser responsáveis pela operação, manutenção e melhorias no sistema liberado.

[1] Figura retirada do Rational Unified Process Tutorial [22]

Segundo [24], a iniciação e a elaboração abrangem as atividades de engenharia do ciclo de vida do desenvolvimento. A construção e a transição constituem sua produção. Nossa proposta para a reutilização de padrões se concentra apenas nas fases de iniciação e elaboração.

Como mostrado na Figura 4, em cada fase podem ocorrer várias iterações. Uma iteração representa um ciclo completo de desenvolvimento. O gráfico mostra como a ênfase varia através do tempo. Por exemplo, nas iterações iniciais, é dedicado mais tempo aos requisitos. Já nas iterações posteriores, é gasto mais tempo com implementação.

5.2. Integração

A Figura 5 ilustra um diagrama de atividades UML que representa a proposta de integração dos mecanismos de busca e aplicação de padrões armazenados no repositório com o RUP. O repositório de padrões aparece interagindo com as fases de iniciação e elaboração do RUP de acordo com os padrões efetivamente aplicados.

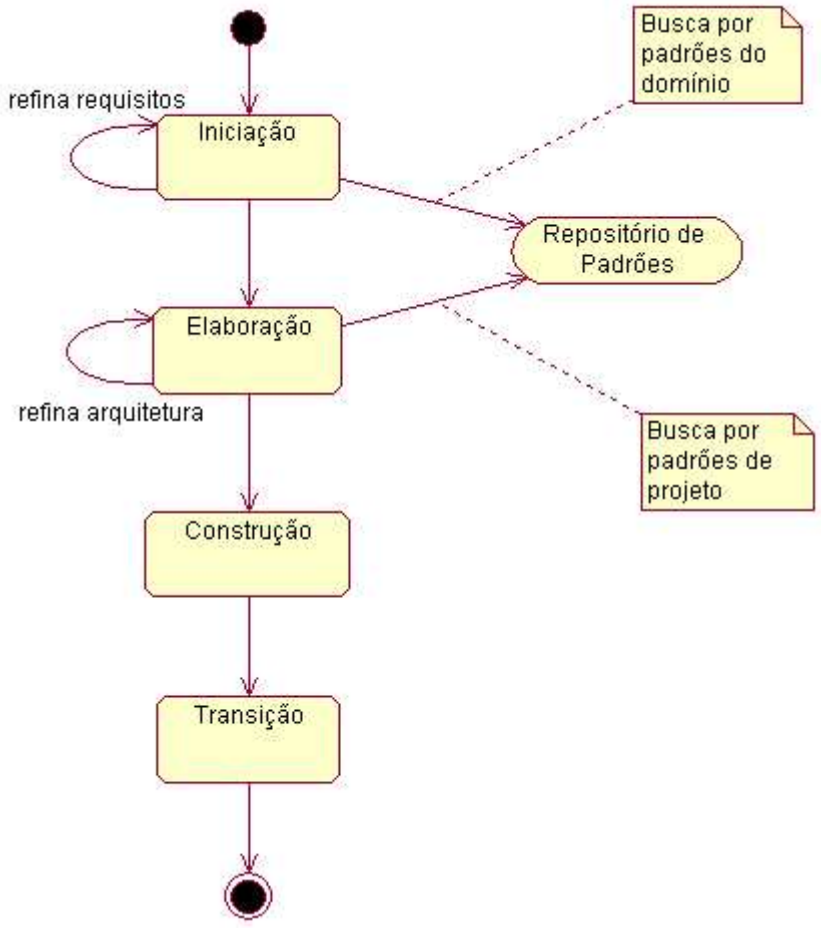

Figura 5 - Procedimento de busca e aplicação de padrões utilizando o RUP

Na fase de iniciação do RUP, o objetivo é estabelecer o escopo do projeto e as condições limite, incluindo uma visão operacional e critérios de aceitação [21]. Nessa fase,

palavras chave do domínio da aplicação são utilizadas para efetuar a primeira busca de padrões. Uma busca no repositório de padrões (veja Seção 4.1.) é realizada com o foco em padrões específicos para o domínio da aplicação. Os padrões, que corresponderem à busca realizada, são capturados e visualizados. De acordo com os padrões capturados, eles podem facilitar o entendimento e a implementação dos requisitos funcionais e não funcionais da aplicação que está sendo desenvolvida.

Na fase de elaboração do RUP, a arquitetura para suportar a solução da aplicação é proposta pela equipe de desenvolvimento. Nesse momento, uma busca no repositório de padrões com o foco em padrões genéricos para soluções de projeto é realizada. Os padrões que correspondem à busca são capturados. O projeto da aplicação pode ser refinado de acordo com os padrões selecionados. Na nossa proposta, as fases de construção e transição do RUP não sofrem nenhuma alteração.

6. Conclusão

Uma das motivações para a realização deste trabalho foi a possibilidade de facilitar a busca de padrões de software e a aplicação dos mesmos nas fases iniciais de desenvolvimento. A importância da reutilização de padrões foi inicialmente investigada através de um estudo de sistemas sendo desenvolvidos no Instituto Atlântico. Em seguida, as dificuldades encontradas na reutilização de padrões serviram de entrada para a especificação dos requisitos e posterior modelagem do repositório de padrões de software.

Este artigo então propõe um procedimento para busca em um repositório de padrões de software e a aplicação dos mesmos no desenvolvimento de sistemas utilizando o RUP. A abordagem apresentada é composta de duas fases de busca de padrões que interagem com as fases de iniciação e elaboração do RUP. Na primeira fase (i.e., fase de iniciação do RUP), a busca dos padrões é realizada com foco em padrões específicos para o domínio da aplicação que está sendo desenvolvida. Na segunda fase (i.e., fase de elaboração do RUP), padrões genéricos para soluções de projeto são selecionados. O mecanismo de busca dos padrões nestas fases utiliza um repositório que possui um catálogo dos prováveis padrões de insumo para as fases do RUP. É permitido ao usuário realizar uma busca sobre todos os padrões catalogados no repositório por meio de palavras chave.

É importante mencionar que o repositório de padrões ainda está em fase de implementação. Como discutido anteriormente, a especificação dos requisitos e a modelagem dos dados já foram feitos. Atualmente, estão sendo realizados estudos voltados para o levantamento de técnicas e algoritmos de busca disponíveis que poderão ser utilizados com eficiência no repositório. Como os templates dos padrões a serem armazenados variam de um padrão para outro, o armazenamento de dados do repositório está utilizando um servidor de banco de dados que suporta uma linguagem apropriada para trabalhar com tipos de dados não estruturados que é XML – eXtensible Markup Language.

7. Referências

[1] Borland JBuilder. Disponível em: http://www.borland.com/jbuilder. Acessado em: 03/07/2003.

[2] Borland Together. Disponível em: http://www.borland.com/together. Acessado em: 03/07/2003.

[3] Budinsky, F., Finnie, M., Vlissides, J., and Yu, P.S., "*Automatic Code Generation From Design Patterns*". IBM Research Journal, Vol. 35, No. 2 , 1996 . Disponível em http://www.research.ibm.com/journal/sj/352/budinsky.html.

[4] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M., *Pattern-Oriented Software Architecture*, John Wiley and Sons, New York, NY, 1996.

[5] Chambers, C., Harrison, W. e Vlissides, J.M., *A debate on language and tool support for design patterns*. In Proceedings In 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, Pages: 277 – 289, ACM Press, 2000.

[6] CodePro Studio. Disponível em: http://www.instantiations.com/codepro/ws/docs/default.htm. Acessado em: 10/06/2003.

[7] Coplien, J. O., Software Patterns, SIGS books and Multimedia, June 1996.

[8] Conte, A., Freire, J., Giraudin., J., Hassine, I., Rieu, D. A tool and a formalism to design and apply patterns. SugarLoafPLoP 2002.

[9] Core J2EE Pattern Catalog. Available at http://java.sun.com/blueprints/corej2eepatterns/. Acessado em: 03/07/2003.

[10] DPT-Tool. University of Technology Munich/Departement of CS (Informatik), Disponível em: http://dpt.kupin.de/. Acessado em: 03/07/2003.

[11] Eclipse Project. Disponível em: http://www.instantiations.com/codepro/ws/docs/default.htm. Acessado em: 02/07/2003.

[12] Fowler, M., *Analysis Patterns: Reusable Object Models*, Addison-Wesley, Reading, MA, 1997.

[13] Gamma E., Helm R., Johnson R., Vlissides J., "*Design Patterns: Element of Reusable Object-Oriented Software "*, 1995.

[14] IBM Websphere Studio. Disponível em: http://www-3.ibm.com/software/info1/websphere. Acessado em: 02/07/2003.

[15] IntelliJ IDEA. Disponível em: http://www.intellij.com/idea/. Acessado em: 30 de abril de 2003.

[16] JRefactory. Disponível em: http://jrefactory.sourceforge.net/csrefactory.html. Acessado em: 03/07/2003.

[17] Kroth, E., Pfanfeseller, M., *Uma ferramenta de apoio ao desenvolvimento de software baseado em componentes*. XV Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas, Rio de Janeiro-RJ, outubro, 2001.

[18] Meszaros, G., Doble, J., "*A Pattern Language for Pattern Writing*," PatternLanguage of Program Design 3, edited by Robert C. Martin, Dirk Riehle, and Frank Buschmann, Addison-Wesley (Software Patterns Series), 1997.

[19] ModelMaker. Disponível em: http://www.modelmakertools.com. Acessado em: 03/07/2003.

[20] Pattern Languages of Program Design I, II, III & IV; Patterns from the PLoP Conference at Allerton Park in Illinois, US and EuroPLoP in Europe; Addison-Wesley, 1994-95-96-98.

[21] Pollice, Gary. Using the Rational Unified Process for Small Projects: Expanding Upon extreme Programming. Rational Software White Paper.

[22] Rational Unified Process Tutorial. Versão 2002 05 00.

[23] Rising, Linda, *The Pattern Almanac 2000*, Software Pattern Series, Addison-Wesley, 2000. ISBN 0-201-61567-3.

[24] Rumbaugh, J., Booch, G., Jacobson, I. *UML Guia do Usuário*. Editora Campus. 2000.

[25] Sommerville, Ian. *Software Engineering*, 6th Edition, Addison-Wesley Publishers Ltd., 2001. ISBN 0-201-39815-X.

[26] Sullivan, D., "*A Webmaster's Guide to Search Engines*", disponível em http://calafia.com/.

Acessado em: 01/07/2003.

[27] UMLStudio. Disponível em: http://www.pragsoft.com. Acessado em: 10/06/2003.