UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA DE SUPORTE A REUSO DE CASOS DE USO

MIRIAM RAMOS MARTINS

BLUMENAU
2006

2006/1-33

MIRIAM RAMOS MARTINS

FERRAMENTA DE SUPORTE A REUSO DE CASOS DE USO

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Everaldo Artur Grahl - Orientador

BLUMENAU
2006

2006/1-33

FERRAMENTA DE SUPORTE A REUSO DE CASOS DE USO

Por

MIRIAM RAMOS MARTINS

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Everaldo Artur Grahl, Mestre – Orientador, FURB
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB
Membro:	Prof. Fabiane Barreto Vavassori Benitti, Doutora – FURB

Blumenau, 14 de junho de 2006

Dedico este trabalho as pessoas que sempre estiveram comigo, em especial a minha família e meu namorado pelo apoio em todos os momentos.

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

À minha família e namorado.

Ao meu orientador.

Nada lhe posso dar que já não exista em você mesmo. Não posso abrir-lhe outro mundo de imagens, além daquele que há em sua própria alma. Nada posso lhe dar a não ser a oportunidade, o impulso, a chave. Eu o ajudarei a tornar visível o seu próprio mundo, e isso é tudo.

Hermann Hesse

RESUMO

Este trabalho teve como objetivo o desenvolvimento de uma ferramenta para a elaboração de casos de uso para serem reutilizados. Os casos de uso são armazenados em um repositório, para que possam ser recuperados e reusados facilmente. Além disso, foi definido um mecanismo de armazenamento, no qual é possível realizar buscas inteligentes através de uma semântica estabelecida, resultando em modelos de casos de uso pronto para o reuso.

Palavras-chave: Reutilização de requisitos. Casos de uso. Repositório. Busca semântica.

ABSTRACT

This work took the development of a tool as an objective for the preparation of use cases in order that they were reused. The use cases must be stored in a repository, so that they can be recovered and reused easily. Besides, there was defined a mechanism of storage, in which it is possible to carry out intelligent searches through an established semantics, turning in models of use cases ready for the reuse.

Key-words: Requirements reuse. Use cases. Repository. Semantic search.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

CORBA - *Common Object Request Broker Architeture*

DTD - *Document Type Definition* ou Definição de Tipo de Documento

MOF - *Meta Object Facility*

OMG - *Object Management Group* ou Grupo de Gerenciamento de Objetos

RUP - *Rational Unified Process* ou Metodologia do Processo Unificado

UML – *Unified Modeling Language* ou Linguagem de Modelagem Unificada

XMI – XML *Metadata Interchange*

XML - eX*tensible Markup Language* ou Linguagens de *Markup* Extensíveis

SUMÁRIO

1 INTRODUÇÃO

Na literatura existem diversas definições para o termo reutilização. Originalmente proposto, durante a conferência sobre engenharia de software da Organização do Tratado do Atlântico Norte (OTAN), este termo foi inicialmente definido como a criação de uma biblioteca de componentes para serem simplesmente reutilizados (MCILROY, 1969). Esta proposição foi uma resposta ao problema intitulado “Crise do Software” definido por Naur (1968), alegando a falta de capacidade demonstrada pela indústria de software para desenvolver sistemas de software confiáveis, flexíveis e de baixo custo.

Para Oliveira (2001, p. 56) o processo de reutilização é marcado pela execução de um conjunto de atividades, desde a busca por um artefato útil, passando por uma verificação de compatibilidade, até chegar à sua efetiva utilização. Em grande parte, ambientes voltados à reutilização não abordam as duas atividades iniciais, devido incapacidade de se obter uma especificação precisa das semânticas envolvidas que possa ser benéfica ao processo como um todo.

No cenário atual, a sobrevivência das organizações no mercado depende de sua competitividade que, hoje, é função direta da produtividade. Sob esta perspectiva, significa dizer que o dinamismo e a crescente competitividade no mundo dos negócios, fazem com que empresas preocupem-se cada vez mais com a qualidade e produtividade no processo de desenvolvimento de software. Nesse sentido, o reuso de requisitos é uma alternativa interessante, pois apresenta benefícios significativos para o processo de desenvolvimento de software como produtividade, qualidade e redução de custos. Identificar requisitos similares ou até mesmo idênticos, os quais são repetidamente desenvolvidos, leva à tentativa de reutilizar requisitos já existentes.

A reutilização pode ser abordada como uma forma de obter requisitos mais confiáveis, uma vez que já foram implementados previamente em outros sistemas. Mesmo sendo confiáveis, para que sejam reutilizados da maneira como foram projetados, é necessário compreender suas características, funcionalidade e comportamento.

No desenvolvimento de software orientado a objeto, usar casos de uso para capturar requisitos funcionais é uma prática comum. Eles são suportados por linguagem de modelagem como a *Unified Modeling Language* (UML) e processos de desenvolvimento como *o Rational Unified Process* (RUP). Uma maneira de mostrar às organizações como reduzir os custos na construção de casos de uso, é a prática de reuso de casos de uso, mas para serem reutilizados de uma forma eficaz e precisa, é necessário saber como construir casos de uso reutilizáveis.

Quando o tamanho e a complexidade dos sistemas de software crescem, aumenta o interesse em desenvolver casos de uso baseados em modelos reusáveis. Os principais benefícios pretendidos com os modelos reusáveis, são diminuição de custos e tempo na especificação de requisitos.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver uma ferramenta de apoio a documentação e recuperação de casos de uso para serem reutilizados.

Os objetivos específicos do trabalho são:

a) definir um processo para elaboração de casos de uso reutilizáveis;

b) desenvolver um repositório de casos de uso, para gerenciar a reutilização;

c) armazenar casos de uso no formato XML *Metadata Interchange* (XMI);

d) elaborar mecanismo de busca semântica em arquivos XMI;

e) gerar modelos de casos de uso, através do resultado da busca.

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizado em quatro capítulos. No segundo capítulo é descrita a fundamentação teórica utilizada para embasar este trabalho, destacando a reutilização de requisitos, analogia, casos de uso, padrão XMI E mecanismo de busca. O capítulo é finalizado com os trabalhos correlatos.

No terceiro capítulo é apresentada a especificação e implementação da ferramenta.

O quarto capítulo contém a conclusão do trabalho, juntamente com sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados alguns aspectos teóricos relacionados ao trabalho, tais como: reutilização de requisitos, analogia, casos de uso, padrão XMI e mecanismo de busca. Na última seção são apresentados alguns trabalhos correlatos.

2.1 REUTILIZAÇÃO DE REQUISITOS

A especificação de requisitos é a primeira fase do ciclo de vida do desenvolvimento de software. No entanto, os requisitos mudam durante o desenvolvimento e evoluem depois que o sistema já está em operação. Um dos grandes desafios é a obtenção de requisitos que sejam os mais precisos possível para possibilitar um bom entendimento do que é desejado e esperado do produto final do processo (SOMMERVILLE e KOTONYA, 1997). O documento de requisitos é o meio através do qual é possível descrever as restrições quanto às características do produto e quanto ao processo de desenvolvimento, a interface com outras aplicações, a descrição sobre o domínio e as informações de suporte ao conhecimento do problema (RYAN, 1993).

Para Blaschek (2002, p. 3) é importante que os gerentes de projeto reconheçam que não é possível desenvolver sistemas de qualidade, cumprir prazos e custos e atender às expectativas dos usuários sem ter um processo de desenvolvimento de requisitos definido, compreendido e utilizado por todos os desenvolvedores.

Portanto, o reuso de requisitos tem por objetivo aumentar a produtividade, diminuir o custo de desenvolvimento e promover a integração e comunicação entre projetos. Além disso, facilita o aprendizado de novas áreas de conhecimento para equipes sem experiência na aplicação a ser desenvolvida.

A reutilização bem sucedida de especificações permite melhorar a produtividade no desenvolvimento de software dando aos desenvolvedores um começo mais rápido durante a análise de requisitos. Ela pode ainda beneficiar a engenharia de requisitos pela melhora da qualidade nas especificações resultantes. Finalmente, ela pode enriquecer a base de conhecimento própria do engenheiro de software, fornecendo a experiência necessária para entender e resolver problemas análogos (PIMENTA, 1998).

2.2 REUTILIZAÇÃO DE REQUISITOS POR ANALOGIA

Existe uma tendência de se explorar a reutilização nas fases iniciais do desenvolvimento de software. Esta tendência se justifica pela grande importância dada a estas fases (NEIGHBORS, 1994). Entretanto, os produtos das fases iniciais são fortemente relacionados com o domínio da aplicação, fazendo com que a reutilização de especificações, de modelos de requisitos ou estratégias de projeto só possa ser realizada com sucesso entre aplicações da mesma família, ou seja, aplicações que compartilhem requisitos e restrições. A noção de domínio é, pois, fundamental à reutilização nas fases iniciais (ZIRBES, 1995).

Para Pimenta (1998, p.24) domínio na engenharia de software pode ser entendido como uma família de sistemas que possuem características semelhantes, e que, justamente por isto, um núcleo bastante significativo de requisitos.

Analogia é um paradigma usado para reutilizar especificações de sistemas análogos, a resolução de problemas análogos consiste na transferência de conhecimento de episódios de problemas resolvidos anteriormente, para novos problemas, que compartilhem aspectos significativos com a experiência anterior, usando transferência do conhecimento a fim de construir soluções para os novos problemas. O poder da analogia é seu potencial de absorver conhecimento de um domínio e aplicá-lo a um outro domínio (MAIDEN; SUTCLIFFE,

1992).

A analogia é um conceito amplamente utilizado em diversas áreas da Ciência da Computação, como por exemplo, raciocínio baseado em casos (KEANE, 1994), paradigmas de aprendizado de máquina, gerenciamento de sistemas e engenharia de software.

Para Maiden e Sutcliffe (1992) o processo de reutilização usando a analogia pode ser dividido em duas etapas principais: a primeira atividade a ser realizada procura identificar um conjunto de domínios abstratos que representem famílias de sistemas com características semelhantes; uma segunda etapa consiste no uso do raciocínio por analogia, ou seja, reconhecer a semelhança entre o problema e os domínios abstratos, compreender esta semelhança, e por último, transferir conhecimento. A segunda etapa, raciocínio por analogia, consiste de três passos principais: recuperação dos componentes, seleção do componente mais adequado e adaptação de componentes. Na prática, isto pressupõe a existência de uma biblioteca, através da qual seja possível a identificação de similaridades e a conseqüente seleção dos recursos reutilizáveis adequados. A organização desta biblioteca deve facilitar a seleção. Uma das maneiras mais simples e naturais de se obter esta facilidade é organizar o catálogo através de domínios ou classes de recursos. A pesquisa pelo modelo similar de um sistema, será então realizada entre os modelos integrantes da classe a qual o novo sistema pertence (ZIRBES, 1995).

A utilização bem sucedida da reutilização de especificações pode ajudar a superar as dificuldades encontradas por engenheiros de software inexperientes durante os estágios iniciais do desenvolvimento de software. Especificações análogas podem fornecer aos engenheiros modelos de domínio relevantes com fronteiras semelhantes para auxiliar na definição do escopo do problema, também podem fornecer cenários alternativos para avaliar novas especificações.

2.3 CASOS DE USO

Os casos de uso podem ser utilizados durante a análise e especificação dos requisitos para descrever a funcionalidade do sistema. Casos de uso representam a interação do sistema com seus usuários. O conceito foi elaborado por Jacobson et al. (1992), e posteriormente incorporado à UML. Desde então, são amplamente utilizados na modelagem dos requisitos de software.

Caso de uso é um conceito amplamente difundido e utilizado para a documentação e o desenvolvimento de requisitos. Segundo o RUP (2003), caso de uso é uma descrição de comportamento do sistema em termos de seqüências de ações. Um caso de uso deve produzir um resultado de valor observável para um ator. Ele contém todos os fluxos de eventos referentes à produção do "resultado de valor observável". Mais formalmente, um caso de uso define um conjunto de instâncias de casos de uso ou cenários. O CMMI (2002) indica que casos de uso podem ser usados na elicitação e análise de requisitos para estabelecer os cenários operacionais do sistema. Ou seja, além de representar os requisitos, os casos de uso também descrevem uma solução em alto nível.

Casos de uso são apenas uma pequena parte do esforço total de captura de requisitos. Eles são uma parte central desse esforço, agindo como um núcleo ou eixo que liga definições de dados, regras de negócio, projeto de interface com usuário, o modelo de domínio do negócio, e assim por diante (COCKBURN, 2005).

Segundo Cockburn (2005) podem-se destacar os objetivos principais dos casos de uso como:

a) delimitação do contexto de um sistema;

b) documentação e o entendimento dos requisitos;

c) descrição dos requisitos funcionais;

d) principal saída da etapa de especificação de requisitos;

e) principal entrada da etapa de análise.

Os objetivos secundários dos casos de uso seriam:

a) facilitar a comunicação entre os *stakeholders*[1];

b) servir de base para a definição do cronograma;

c) auxiliar na elaboração dos casos de teste.

Cada caso de uso deve ser definido através da descrição narrativa das interações que ocorrem entre o(s) elemento(s) e o sistema. A UML não define o formato e o grau de abstração a serem utilizados na descrição de um caso de uso. Consequentemente, há vários formatos de descrição propostos na literatura, assim como vários são os graus de abstração utilizados.

2.3.1 Formato

Segundo Bezerra (2002) alguns tipos de formato freqüentemente utilizados são a descrição contínua, a descrição numerada e a descrição particionada. Esses tipos de narrativas são apresentados a seguir:

a) descrição contínua: a narrativa é feita através de um texto livre;

b) descrição numerada: a narrativa é descrita através de uma série de passos numerados;

c) descrição particionada: tenta prover alguma estrutura à descrição de casos de uso. Neste estilo, a seqüência de interações entre o ator e o sistema é particionada em duas colunas, uma para o ator e outra para o sistema. Essa forma de estruturação da narrativa tem o objetivo de separar as ações do ator e as reações do sistema.

Sobre o grau de abstração da narrativa de um caso de uso diz respeito à existência ou

[1] É um ator externo com o direito de ter seus interesses protegidos pelo sistema, e satisfazer os seus interesses requer que o sistema tome atitudes específicas. Diferentes casos de uso podem ter diferentes *stakeholders*.

não de menção à tecnologia a ser utilizada na descrição desse caso de uso. Um caso de uso pode ser real ou essencial. Um caso de uso essencial é abstrato e não faz menção à tecnologia a ser utilizada. Diferentemente, em um caso de uso real, as descrições das interações citam detalhes da tecnologia a ser utilizada na implementação deste caso de uso. A descrição em um grau de abstração real se compromete com a solução de projeto a ser utilizada para implementar o caso de uso. Casos de uso reais são também chamados de casos de uso concretos (BEZERRA, 2002, p. 47).

2.3.2 Atores

Na terminologia da UML, qualquer elemento externo que interage com o sistema é denominado ator. O termo “externo” indica que atores não fazem parte do sistema. O termo “interage” significa que um ator troca (envia e/ou recebe) informações com o sistema.

Para todas as categorias de atores, os casos de uso representam alguma forma de interação, no sentido de troca de informações, entre o sistema e o ator. Um caso de uso pode ser visto como um modo específico de utilização de alguma funcionalidade. Normalmente um agente externo inicia a seqüência de interações com o sistema, ou um evento acontece para que o sistema responda.

Um ator pode participar de muitos casos de uso, do mesmo modo, um caso de uso pode envolver vários atores, o que resulta na classificação dos atores em primários ou secundários. Um ator primário é aquele que inicia uma seqüência de interações de um caso de uso. São os agentes externos para os quais o caso de uso traz benefício direto. Atores secundários supervisionam, operam, mantêm ou auxiliam na utilização do sistema. Esses atores existem apenas para que os atores primários possam utilizar o sistema (BEZERRA, 2002, p. 51).

2.3.3 Cenários

Geralmente um caso de uso tem diversas maneiras de ser realizado. Um cenário é a descrição de uma das maneiras pelas quais um caso de uso pode ser realizado. Um cenário também é chamado de instância de um caso de uso. Normalmente há diversos cenários para um mesmo caso de uso. Um uso importante dos cenários está no esclarecimento e no entendimento dos casos de uso dos quais eles são instanciados.

Cada cenário começa com uma condição de acionador que indica quando ele é executado e vai até mostrar a conclusão ou o abandono do seu objetivo (COCKBURN, 2005, p. 94). Portanto os cenários descrevem o caso de uso numa linguagem fácil de entender e validar para todas as pessoas relacionadas com o projeto, motivando-as a discutir e participar, obtendo assim um maior retorno sobre o andamento do trabalho.

Cada cenário, conforme exemplo no Quadro 01, descreve textualmente cada um dos elementos separadamente ou seqüência que caracteriza o comportamento do ator e as respostas do sistema.

Casos de uso: Validar cliente
Cenário principal: o sistema solicita ao cliente (do banco) a sua senha. O cliente fornece os números através do teclado. O cliente confirma a senha pressionando a tecla entre. O sistema checa este número e verifica se ele é válido.
Cenário de exceção 1: o cliente pode cancelar a transação a qualquer momento pressionando o botão cancelar, retornando ao cenário principal. Não é feita nenhuma mudança na conta do usuário.
Cenário de exceção 2: o cliente pode corrigir a senha a qualquer momento antes de confirmar com a tecla entre.
Cenário de exceção 3: se o cliente fornece um número de senha inválido a transação é reiniciada. Se isto acontecer três vezes seguidas, o sistema cancela a transação e bloqueia por até uma hora.

Quadro 01 – Cenário de Casos de uso

O tipo de cenário principal é o mais usado, que normalmente é um conjunto de passos, os casos de uso também podem conter cenários alternativos que contêm variações do tema principal e cenários de exceções, que acontece quando as coisas não correm bem (COCKBURN, 2005).

2.3.4 Relacionamentos

Segundo Bezerra (2002) a UML define diversos tipos de relacionamentos no modelo de casos de uso:

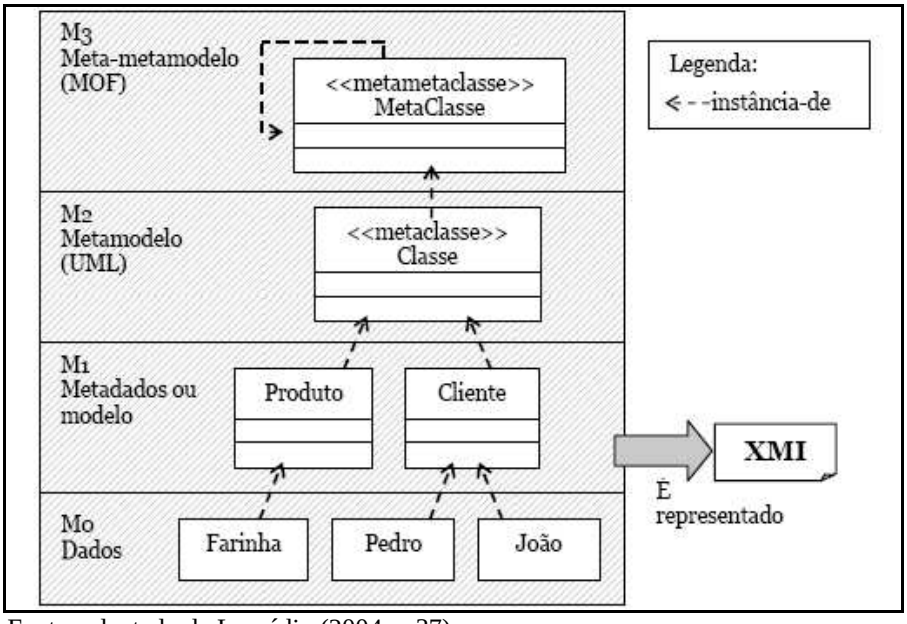
a) comunicação: representa a informação de quais atores estão associados a que casos de uso. O fato de um ator estar associado a um caso de uso significa que esse ator interage com o sistema;
b) inclusão: existe somente entre casos de uso. Quando dois ou mais casos de uso incluem uma seqüência comum de interações, essa seqüência comum pode ser descrita em um outro caso de uso. Isso evita a descrição de uma seqüência de interações mais de uma vez;
c) extensão: utilizado para modelar situações em que diferentes seqüências de interações podem ser inseridas em um caso de uso, chamado de caso de uso estendido;
d) generalização: permite que um caso de uso herde características de um caso de uso mais genérico, este último normalmente chamado de caso de uso base. O caso de uso herdeiro pode especializar o comportamento do caso de uso base.

2.4 O PADRÃO XMI

Em virtude dos problemas de integração entre ferramentas de modelagem, o *Object*

Management Group (OMG) definiu uma proposta para facilitar o intercâmbio de metadados. Essa proposta é o padrão XMI, e consiste no uso da linguagem *Extensible Markup Language* (XML) para representar metadados, como por exemplo modelos UML. No nível conceitual, o XMI é baseado em outro padrão da OMG chamado *Meta Object Facility (*MOF). O MOF é um padrão para definição de interfaces de programação *Common Object Request Broker Architeture* (CORBA) para repositórios de modelos, mas é também um padrão para descrever metamodelos (RIBEIRO; FLORENTINI, 2000, p. 4).

A Figura 01 mostra a relação entre MOF, UML, e XMI. São definidos quatro níveis: o primeiro nível (M0) corresponde aos dados propriamente ditos. O segundo nível (M1) corresponde aos metadados, ou modelo. São os dados que descrevem os dados. O terceiro nível (M2) é o metamodelo para definição de modelos. A especificação UML é um exemplo de metamodelo. O padrão MOF encontra-se no quarto nível (M3). Nesse nível estão os modelos que definem metamodelos (LUCRÉDIO, 2004).



Fonte: adaptado de Lucrédio (2004 p. 37).

Figura 01 – Relação entre MOF, UML e XMI

No Quadro 02 apresenta-se um exemplo de um trecho de código XMI que mostra o elemento XMI.extensions usado pela ferramenta de modelagem Enterprise Architect (SPARX SYSTEMS, 2006) para representar o modelo EAModel.scenario, contendo os cenários

“Efetua *Login*”, “*Login* Inválido” e “*Login*/Senha em branco”.

```
<XMI.extensions xmi.extender="Enterprise Architect 2.5">
      <EAModel.scenario>
            <EAScenario name="Efetua Login" type="Principal"
            description="1º Passo O sistema apresenta uma página ..." />
            <EAScenario name="Login Inválido" type="Exceção"
            description="No 3º Passo, o login ou a senha não puderem ser..."/>
            <EAScenario name="Login/Senha em branco" type="Exceção"
            description="No 2º Passo, a conta ou a senha estiver em..."/>
      </EAModel.scenario>
</XMI.extensions>
```

Quadro 02 – Exemplo resumido de um arquivo XMI

Utilizando XMI, é possível representar modelos UML (a modelagem de um domínio ou o projeto de componentes de software, por exemplo). A representação em XMI nada mais é do que a representação em XML, estruturada através de uma *Document Type Definition* (DTD) específica para representar modelos. A DTD funciona para os documentos XML como uma gramática, onde são definidos todos os tipos de marcação que podem ser utilizados. A DTD permite que sejam incluídas restrições aos documentos XML, que podem depois ser verificadas pelo processador, a fim de determinar a validade ou não de um documento. Portanto, um documento XMI é um documento XML (TITEL, 2003). No Quadro 03 apresenta-se um exemplo de DTD quee mostra um elemento “*EAScenario*”, o símbolo “*” indica nenhuma ou mais ocorrências. O elemento “*EAScenario*” contem quatro subelementos (*EAName, EAType, EADescription* e *UML:ModelElement.taggedValue*). Para definir os atributos (name, type, description e subject) do elemento “*EAScenario*” é utilizado a *tag ATTLIST*.

```
<!ELEMENT EAScenario      (EAName |EAType |EADescription|
                              UML:ModelElement.taggedValue )*>

<!ATTLIST EAScenario
            name                CDATA #REQUIRED
            type                CDATA #REQUIRED
            description         CDATA #IMPLIED
            subject             IDREFS #REQUIRED
>
```

Quadro 03 – Exemplo de uma DTD

Uma das vantagens do XMI é a interoperabilidade entre ferramentas. Num ambiente de

desenvolvimento, existem diversas ferramentas para auxiliar os Engenheiros de Software. Na maioria dos casos, essas ferramentas trabalham com soluções proprietárias, e dificilmente podem ser integradas. Se cada ferramenta armazenar e recuperar artefatos no formato XMI, elas podem trocar informações sem a necessidade de mecanismos especiais de exportação e importação. Além disso, o escopo da informação descrita em XMI não é fixo, podendo se estender conforme DTDs mais específicas. Portanto, cada ferramenta pode adicionar suas próprias informações, sem prejudicar as já existentes (LUCRÉDIO, 2004).

Outra vantagem é a possibilidade de reuso. Quando se trabalha em diferentes projetos para clientes distintos, o engenheiro de software pode se deparar com ferramentas diferentes em cada cliente. Através da interoperabilidade entre ferramentas obtida com o padrão XMI, é possível, por exemplo, reutilizar os modelos de um projeto A, em um outro projeto B, sem a necessidade de nenhuma conversão.

2.5 REPOSITÓRIO

Elaborar e documentar casos de uso é uma forma eficaz de reutilizar conhecimento sobre o desenvolvimento de sistemas. Para que a reutilização de casos de uso seja realizada de forma eficiente são necessários mecanismos ou ferramentas que auxiliem na automatização do processo de reuso. A construção de um repositório vem ao encontro desta necessidade. O primeiro passo no desenvolvimento de um repositório é a criação de um esquema de classificação para os componentes, pois há a necessidade em armazenar e posteriormente recuperar componente eficientemente (VITHARANA; ZAHEDE; JAIN, 2003, p. 652).

Segundo Hall (1999) é mais importante manter um conjunto restrito de componentes reutilizáveis, de um domínio de aplicações específico, do que manter e gerenciar um grande número de componentes variados. Já Basili e Caldiera (1991) dizem que: “a eficiência do

reuso exige um grande catálogo de objetos reutilizáveis”.

Atualmente, o conceito predominante para o suporte à administração da informação é o de repositório de metadados, ou seja, uma tecnologia capaz de tratar as informações relativas aos dados (metadados), inclusive aquelas relacionadas ao contexto em que eles são utilizados. (CERQUEIRA, 2004. p. 1).

Para Silva e Videira (2001, p. 403) o termo repositório designa o componente da arquitetura das ferramentas que é utilizado como meio de armazenamento de modelos, documentos, ou quaisquer outros artefatos, produzidos por algum dos componentes que completam a arquitetura. Na prática, o papel do repositório pode ser concretizado através de uma base de dados, mas muitos produtos utilizam um simples sistema de gestão de arquivos, alguns com formatos proprietários.

O repositório de uma ferramenta é particularmente relevante, uma vez que facilita a gestão de modelos elaborados, e a respectiva reutilização, disponibilizando para isso mecanismos potentes de pesquisa. Silva e Videira (2001, p. 405) definem que o conteúdo incluirá:

a) modelos de âmbito variado, nomeadamente de diagramas e de documentos, que facilitam a elaboração de artefatos concretos a partir de modelos genéricos;
b) modelos e *frameworks aplicacionais*, a partir dos quais podem ser construídos "esqueletos" de aplicações em função de um conjunto de parâmetros;
c) bibliotecas de objetos, classes e componentes, que para além de eventuais componentes que possam vir inicialmente com as ferramentas, permitem a integração de outros desenvolvidos ao longo do tempo;
d) diagramas diversos que resultam da modelagem do sistema;
e) código fonte, programas executáveis e aplicações empacotadas prontas para distribuir aos usuários finais;

f) arquivos de dados para testes e *scripts* de execução dos mesmos.

O repositório apresenta as funcionalidades típicas de um sistema de gestão de bases de dados, nomeadamente no que diz respeito a:

a) garantia de integridade de dados;

b) compartilhamento de informação;

c) suporte ao trabalho concorrente de vários usuários;

d) facilidades de realização de operações de pesquisa.

O repositório é um componente crítico ao providenciar mecanismos e estruturas de dados para a integração entre as ferramentas, constituindo-se como o meio de armazenamento comum, para todos os artefatos. No entanto, e de modo a garantir o sucesso do repositório, enquanto facilitador do compartilhamento de informação é necessário que sejam definidos adicionalmente os seguintes aspectos:

a) um formato comum para troca de informação descritiva dos artefatos;

b) uma interface comum para assentir e utilizar os artefatos.

2.6 MECANISMO DE BUSCA

A idéia da busca semântica tem como objetivo construir um mecanismo de busca que não faça apenas uma simples pesquisa por palavras, mas que reconheça o significado das palavras pesquisadas no contexto desejado. Dessa forma são exibidos sempre resultados de alta relevância. Neste contexto, a ligação semântica das informações, dá uma gama de possibilidades para mecanismos de recuperação de informações (SOUZA; ALVARENGA, 2004).

A Base de Contextos é uma estrutura que armazena os relacionamentos entre as palavras de um mesmo contexto. Uma implementação possível é estruturar a Base de

Contextos como uma rede semântica, onde os nodos são as palavras e os elos representam relações entre palavras de um mesmo contexto. Entretanto, como pode haver o caso de uma palavra estar relacionada com duas outras em contextos diferentes, há a necessidade de se caracterizar o contexto de cada elo (NAVEGA, 2004).

Ao realizar a busca, a relevância é determinada pela presença dos termos de busca. A fórmula utilizada é simples: cada vez que o termo aparece no documento, soma-se um ponto. Uma lista será então apresentada ao usuário, que poderá consultar diretamente os documentos.

É de certa forma surpreendente a dificuldade que se tem para selecionar material textual relevante a partir de algumas palavras chaves. Em termos computacionais, este problema já foi resolvido há tempos, mas os resultados não parecem ser o que se deseja. Quando se fala de mecanismos de busca por palavra-chave tem-se uma fonte de informação rápida, capaz de colocar na ponta dos dedos milhares de referências de casos de uso no contexto que desejado. Mesmo assim, é comum gastar-se muito tempo selecionando aquilo que realmente interessa. Por isso, essa forma de acesso não é capaz de potencializar o uso da informação disponível eletronicamente. O problema típico que se enfrenta está relacionado à semântica das palavras, ou seja, ao seu significado. Na verdade, há como argumentar que somente agentes inteligentes podem desenvolver uma real semântica do mundo natural. Contudo, deixando de lado essa questão filosófica pode-se encontrar algumas aproximações muito úteis (NAVEGA, 2004).

2.7 TRABALHOS CORRELATOS

Santos (2004) apresenta uma proposta para a reutilização de padrões de software em ferramentas de desenvolvimento, de forma integrada e por demanda, a partir de um formato aberto, o XMI. Além disso, um repositório de padrões de software foi desenvolvido propondo

uma nova abordagem para o armazenamento e busca de diferentes tipos de padrões de software existentes.

Justino (1999) cria um processo de reutilização de especificação estruturada com o auxílio de uma ferramenta. A ferramenta desenvolvida está baseada na técnica da analogia e permite a reutilização de diagramas de fluxo de dados, modelos entidades-relacionamento e dicionário de dados a partir de sistemas já modelados na ferramenta. Normalmente analogias entre especificações concretas permitem realizar um aproveitamento mais efetivo pois os componentes envolvidos são geralmente os mesmos. Isto quer dizer que com o reuso de especificações concretas tem-se poucas customizações a fazer.

Mannion et al. (1999) se propõe a reutilizar requisitos e discriminantes (tipo especial de requisitos que diferencia um sistema de outro) dentro de uma família de aplicações. Cada aplicação da família é construída através da implementação de porções de comportamento específicas para ela. Assim, é possível reutilizar, além dos requisitos separados, requisitos projetados em conjunto para resolver a funcionalidade genérica de um domínio de aplicações.

Rolland, Souveyet e Achour (1998) trabalha com blocos de cenários, que são detalhamento de fluxos básicos de casos de uso, como sendo componentes reutilizáveis, em níveis diferentes de abstrações. O nível de abstração incorre da forma como são descritos os requisitos. De modo geral, os blocos de cenários possuem uma representação desde o nível de requisitos de negócio, onde se tem uma visão mais abrangente e próxima do domínio do problema, até os níveis de requisitos do usuário, onde a visão do sistema reflete a funcionalidade requerida ao software.

3 DESENVOLVIMENTO DO TRABALHO

Neste capítulo são descritos os requisitos, a especificação da ferramenta e sua implementação. Além disso, a operacionalidade da ferramenta e finaliza com uma descrição dos resultados obtidos.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Casos de uso são fundamentalmente uma forma textual, embora possam ser escritos usando fluxogramas, diagramas de seqüência, redes de Petri ou linguagens de programação Cockburn (2005, p. 21). Os casos de usos incluídos no repositório da ferramenta será na forma textual, com ênfase a separação de elementos em cenários (principal e alternativo).

A ferramenta SucReuse criada deverá auxiliar os engenheiros de software a reutilizar e elaborar modelos de casos de uso. O processo para a reutilização se dará em dois momentos. Primeiramente o engenheiro de software deverá através da técnica de analogia levantar casos de uso que estejam inseridos em um mesmo contexto, ou seja, que compartilham conceitos similares. Por exemplo: empréstimos financeiros, e aplicações financeiras, estão relacionados ao mesmo contexto, que se pode classificar como um contexto financeiro. Depois de identificado o contexto o engenheiro irá identificar no cenário do caso de uso as partes fixas e as partes variáveis. As partes variáveis serão marcadas como termo onde deverá ser informado um conceito ou mais para este termo, desta forma tem-se uma ligação semântica entre o termo e os conceitos. Com este processo criou-se uma base de conhecimento para cada modelo de casos de uso adicionado ao repositório da ferramenta. O segundo momento da reutilização acontece quando o usuário da ferramenta ira realizar uma busca para encontrar um modelo que atenda a sua necessidade. As palavras digitadas para realizar a busca serão

inferidas na base de conhecimento do caso de uso para encontrar um modelo que melhor atenda ao usuário. A partir destas necessidades foram levantados os seguintes requisitos funcionais e não funcionais:

a) permitir a inclusão, alteração e exclusão de casos de uso na forma textual (requisito funcional - RF);
b) realizar buscas semântica nos cenários dos casos de uso (RF);
c) efetuar geração de modelos de casos de uso a partir da busca realizada (RF);
d) gerenciar modelos de casos de uso em um repositório (RF);
e) arquivar os casos de uso em um formato de especificação padronizado e aberto, o XMI (requisito não-funcional - RNF).

3.2 ESPECIFICAÇÃO

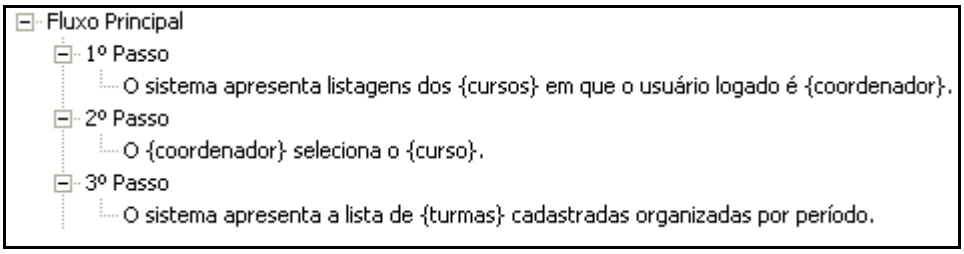
Os requisitos descritos na seção anterior descrevem o que deve ser feito para construir uma ferramenta de elaboração de casos de uso para serem reutilizados. O próximo passo é a especificação de como a ferramenta será desenvolvida. A Figura 02 descreve a arquitetura da ferramenta.

Figura 02 – Estrutura da ferramenta SucReuse

Tem-se o metadados de modelos UML, representado pelo padrão XMI, sendo que o repositório irá armazenar os arquivos XMI importados.

Os casos de uso devem primeiramente ser desenhados em outra ferramenta em forma de diagrama, onde o comportamento associado a cada caso de uso deve ser descrito como um cenário. Os cenários dos casos de uso devem possuir um fluxo principal, podendo ter ou não fluxos alternativos e de exceções.

Na gerência dos casos de uso, será a contextualização do caso de uso, ou seja, o engenheiro de software deverá utilizar estratégias para seleção de domínios análogos e criar um contexto para este domínio. A seguir deve ser analisado o cenário do caso de uso, e definir quais termos devem ser conceituados, para serem utilizados como ligação semântica, conforme mostra o Quadro 04, onde os termos: curso, coordenador, turma, foram marcados, para serem definidos os conceitos possíveis. Por exemplo, curso: pode-se conceituar como palestra e seminário, coordenador: palestrante e seminarista, turma: platéia, ouvinte. Desta forma cria-se uma base de conhecimento para cada caso de uso.



Quadro 04 – Ligação semântica do cenário de caso de uso

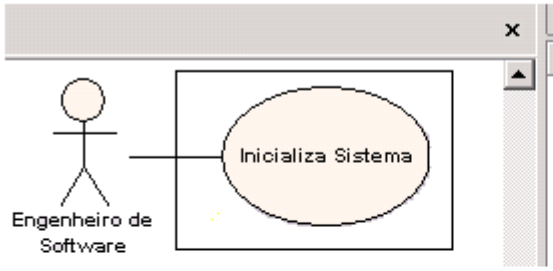
O mecanismo de busca efetuará inferência na base de conhecimento dos casos de uso a partir do escopo da necessidade do usuário ou requisito funcional do problema em questão. Visando maximizar a reutilização, o mecanismo de busca deve recuperar não só casos de uso que atendam exatamente à requisição do usuário, mas também casos de uso “próximos” a estes. Dessa maneira, aumentam-se as chances de reuso, já que mais casos de uso são apresentados ao reutilizador, e a possibilidade de um deles satisfazer suas necessidades é maior. Após a identificação do caso de uso que atenda a sua necessidade, o usuário deve exportar o mesmo para ser efetivamente reutilizado.

Na secção 3.2.1 são apresentados os diagramas de casos de uso da ferramenta. Os

atores envolvidos com a ferramenta são: a ferramenta SucReuse, o engenheiro de software responsável pela especificação dos modelos de casos de uso e o usuário da ferramenta que irá realizar as buscas no repositório para reutilizar os modelos definidos pelo engenheiro de software. O engenheiro de software deve ter domínio do negócio do sistema em desenvolvimento.

3.2.1 Casos de uso

A seguir são apresentados os casos de uso, dando uma idéia das funcionalidades envolvidas na ferramenta, conforme mostrados nos Quadros 05, 06, 07, 08 e 09.



Scenarios

Scenario	Type
Inicializa Repositório	Principal
Cria Repositório	Alternativo
Repositório inconsistente	Exceção
Contexto inconsistente	Alternativo

Scenarios

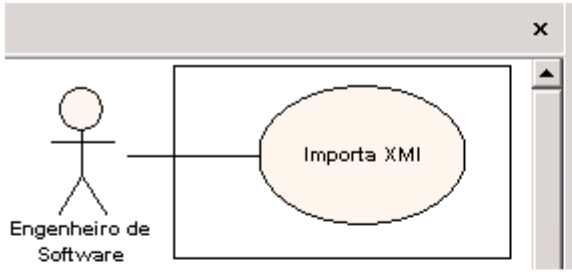
Inicializa Repositório {Principal}.
1º Passo O sistema verifica se já possui um repositório cadastrado no arquivo de configurações dos repositórios.
2º Passo O sistema inicializa o repositório como o padrão da ferramenta.
3º Passo O sistema é inicializado.

Cria Repositório {Alternativo}.
No 1º Passo, caso não exista um repositório cadastrado, o Engenheiro de Software pode optar por criar um repositório.
1.1. A tela de assistente de configuração de repositório é apresentada ao usuário
1.2. O Engenheiro de Software deve escolher a opção Criar um novo repositório
1.3 O Engenheiro de Software deve informar a Localização do repositório.
1.4. O Engenheiro de Software deve informar um ou mais contextos para o repositório, sendo que o primeiro contexto informado será o contexto padrão do repositório.
1.4 O sistema retorna ao passo 3

Repositório inconsistente {Exceção}.
No Passo 1.3, caso não for informado uma localização para o repositório, apresenta mensagem "Informe uma localização "

Contexto inconsistente {Alternativo}.
No Passo 1.4, caso não for informado um contexto para o repositório, apresenta mensagem "Informe um contexto"

Quadro 05 – Diagrama de caso de uso: Inicializa sistema



Scenarios

Scenario	Type
Importa XMI	Principal
Importa Cenário	Alternativo
Formata Cenário	Alternativo

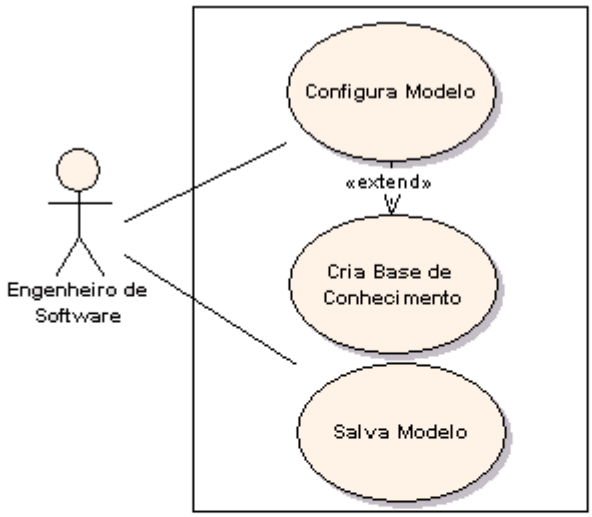
Scenarios

Importa XMI {Principal}.
1° Passo O Engenheiro de Software seleciona o arquivo XMI para importação.
2° Passo O sistema informa a versão do arquivo XMI.
3° Passo O sistema informa a ferramenta de modelagem usada para exportação do arquivo.
4° Passo O Engenheiro de Software confirma a importação do arquivo XMI.
5° Passo O sistema importa o arquivo XMI.
6° Passo O sistema apresenta o casos de uso do arquivo XMI
7° Passo O sistema libera a edição do cenário do caso de uso.

Importa Cenário {Alternativo}.
No 5° Passo, caso os casos de uso do arquivo XMI, possuir cenário.
5.1 O sistema importa o cenário.
5.2 O sistema retorna ao passo 6

Formata Cenário {Alternativo}.
No 5.1 Passo, caso foi importado os cenários do casos de uso.
5.1.1 O sistema formata o cenário conforme a formatação predefinida pelo engenheiro de software.
5.1.2 O sistema retorna ao passo 5.2

Quadro 06 - Diagrama de caso de uso: Importa XMI



Configura Modelo

Scenarios

Selecionar Termos {Principal}.
1° Passo O Engenheiro de Software seleciona o modelo de caso de uso que será configurado para ser armazenado no repositório.
1° Passo O Engenheiro de Software identifica no cenário do caso de uso os termos variáveis para ser marcados.
2° Passo O Engenheiro de Software marca com os abre chaves e fecha chaves o termo escolhido.
3° Passo O sistema salva no cenário o termo marcado.
4° Passo O sistema apresenta no cenário o termo marcado.

Configurar Base {Alternativo}.
No 3° Passo, após marcas os termos caso o Engenheiro de Software desejar configurar a base de conhecimento.
3.1 O Engenheiro de Software seleciona a opção Configurar Base.
3.2. O sistema verifica no modelo de caso de uso todos os termos marcados pelo o Engenheiro de Software.
3.3 O Engenheiro de Software informa um ou mais conceito para cada termo apresentado para ser conceituado.
3.4 O sistema retorna ao passo 4.

Salvar Base {Alternativo}.
No passo 4, caso o Engenheiro de Software opte por criar a base de conhecimento, executa o caso de uso Cria Base de Conhecimento

Cria Base de Conhecimento

Scenarios

Criar Base de Conhecimento {Principal}.
1° Passo O sistema salva no arquivo XMI do modelo de caso de uso todos os termos marcados nos cenários.
2° Passo O sistema salva no Arquivo XMI do modelo de caso de uso os conceitos de cada termo marcado pelo Engenheiro de Software.

Salva Modelo

Scenarios

Salva Modelo {Principal}.
1° Passo O sistema salva o modelo de caso de uso importado na ferramenta no repositório padrão da ferramenta e no contexto padrão do repositório.

Quadro 07 – Diagrama de caso de uso: Base de Conhecimento

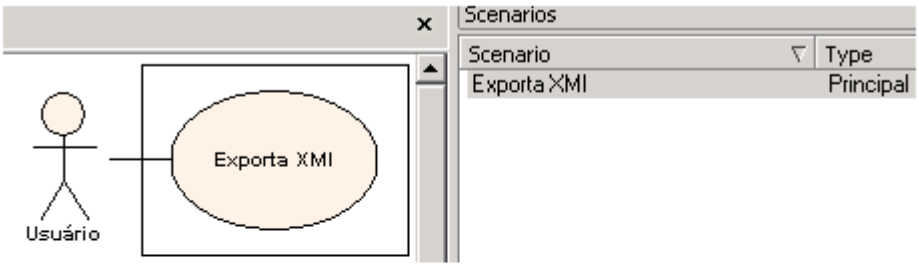
Scenarios

Efetua Busca {Principal}.
1° Passo O sistema mostrará ao usuário uma lista de contextos do repositório.
2° Passo O usuário deve selecionar os contextos em que deseja que a busca deve ser realizada.
3° Passo O usuário deve digitar um escopo da necessidade da busca ou requisito funcional do problema em questão que se deseja encontrar um modelo para reusar.
4° Passo O usuário deve selecionar os conceitos mais importantes na busca.
5° Passo O sistema efetuará inferência na base de conhecimento gerada para cada caso de uso e mostrará ao usuário os casos de uso com os termos de maior relevância relacionados aos conceitos digitados pelo usuário.

Contexto Inconsistente {Exceção}.
No Passo 2, caso não for informado um contexto ou mais para o realizar a busca, apresenta mensagem "Selecione um contexto ou mais "

Escopo Inconsistente {Exceção}.
No Passo 3, caso não for informado um escopo para realizar a busca, apresenta mensagem "Digite o escopo para realizar a busca"

Quadro 08 – Diagrama de caso de uso: Efetuar busca



Scenarios

Exporta XMI {Principal}.
1° Passo O sistema localiza no repositório da ferramenta o arquivo selecionado para ser exportado.
2° Passo O sistema localiza no arquivo do modelo de caso de uso todos os termos entre chaves para ser substituído pelo conceito que foi informado na busca.
3° Passo O sistema faz a substituição dos termos marcados.
4° Passo O sistema exporta o modelo de casos de uso com os termos substituídos para um arquivo XMI, que deve ser importado na ferramenta de modelagem Enterprise Architect.

Quadro 09 – Diagrama de caso de uso: Exporta XMI

3.2.2 Diagrama de Classes

Apresentam-se nesta seção os principais diagramas de classe da ferramenta. Foram divididos em cinco diagramas: inicializa sistema, manipulação do metadados, importa XMI, base de conhecimento, efetuar busca e exporta XMI

A modelagem é baseada no padrão UML de representação de classes para um sistema orientado a objetos. Foi utilizado o Enterprise Architect para o desenvolvimento dos diagramas de classe.

No Quadro 10 tem-se o diagrama de classes que representa a modelagem da inicialização do sistema, onde podem ser visualizadas as quatro classes existentes no mesmo, sendo elas: *TRepository, TFileCfgRepository, TRepositoryLocalization* e *TRepositoryContext.*

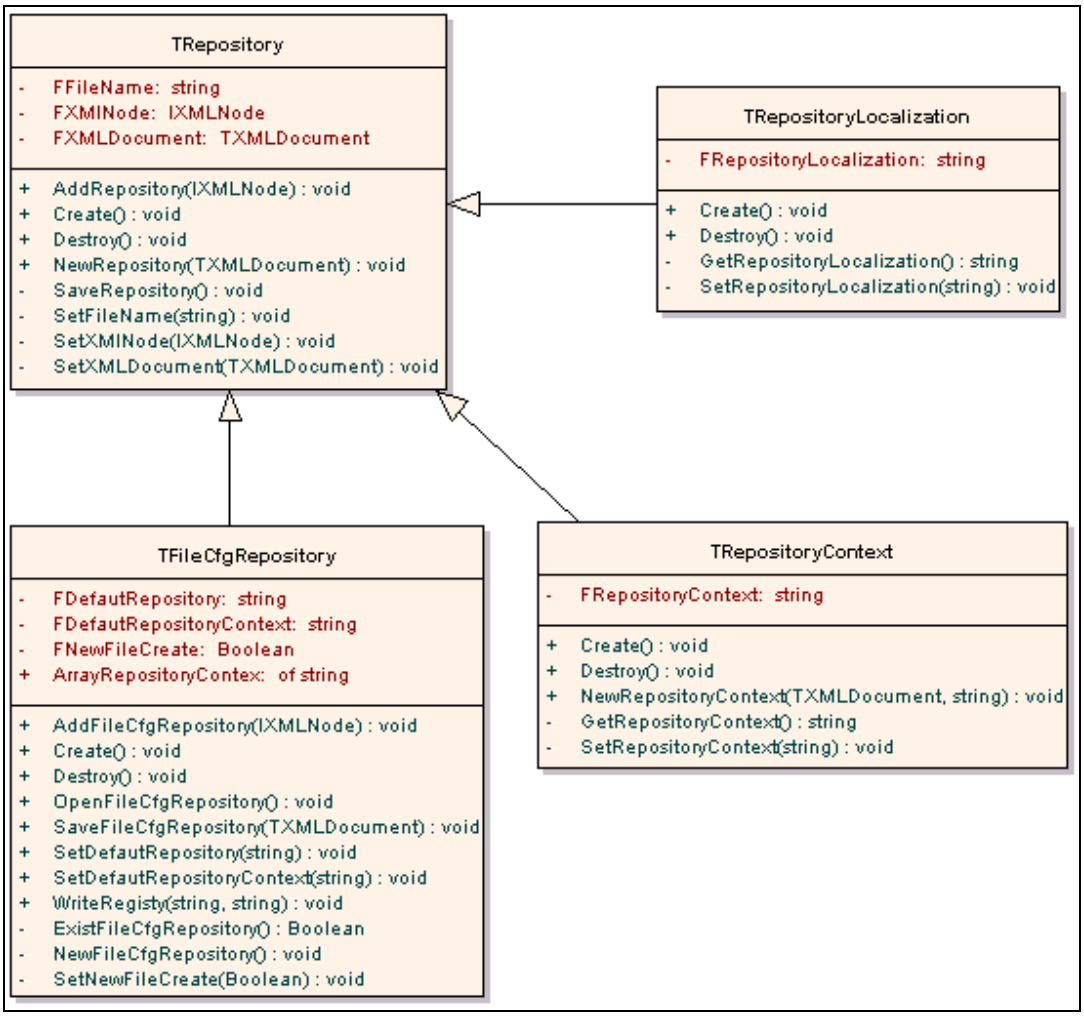
A classe *TRepository* é a classe base para as classes *TFileCfgRepository, TRepositoryLocalization* e *TRepositoryContext,* que possui os seguintes atributos:

a) *FFileName* – possui o caminho completo do arquivo *SucRepository.xml* este arquivo possui a definição dos repositórios cadastrados;

b) *FXMINode* – possui o nodo em que esta sendo feito a escrita ou leitura do arquivo XML;

c) *FXMLDocument*- possui o arquivo XML que está sendo manipulado.

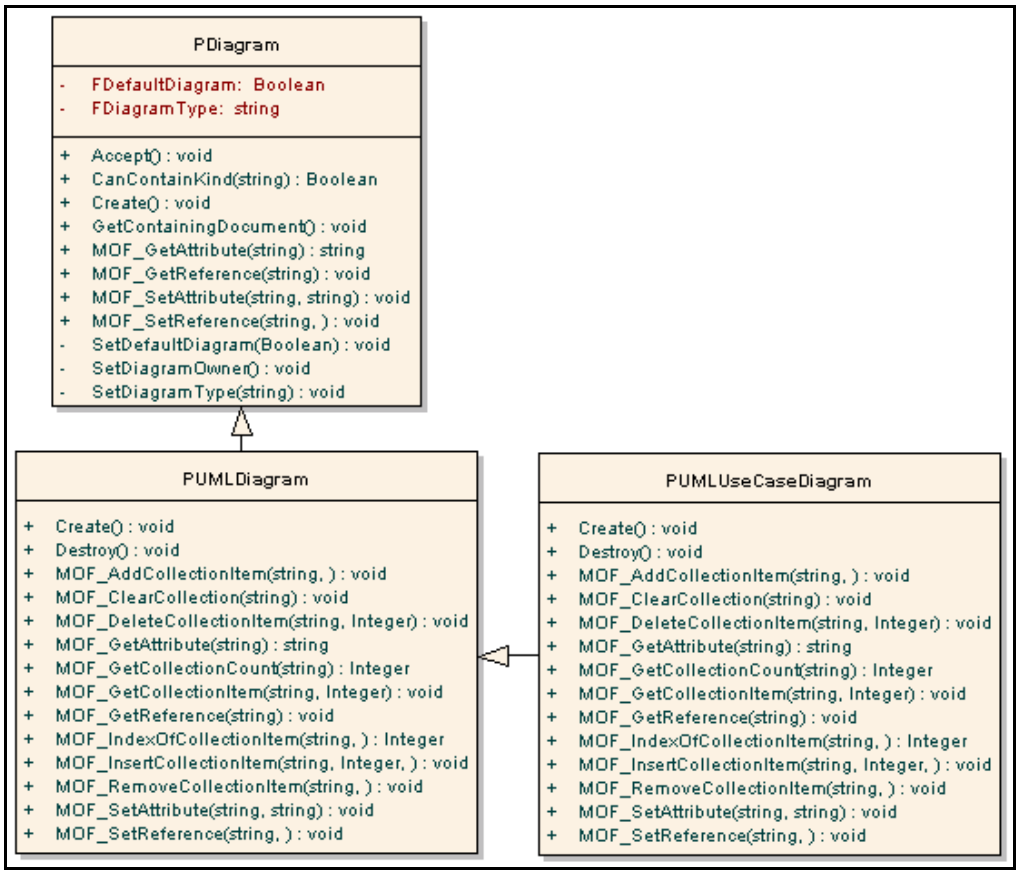
A classe *TFileCfgRepository* é responsável pela escrita e leitura do arquivo de configuração *SucRepository.xml*. As classes *TRepositoryLocalization* e *TRepositoryContext* servem para manipular os nodos *Localization* e *Context* do arquivo XML.



Quadro 10 – Diagrama de classes: Inicializa sistema

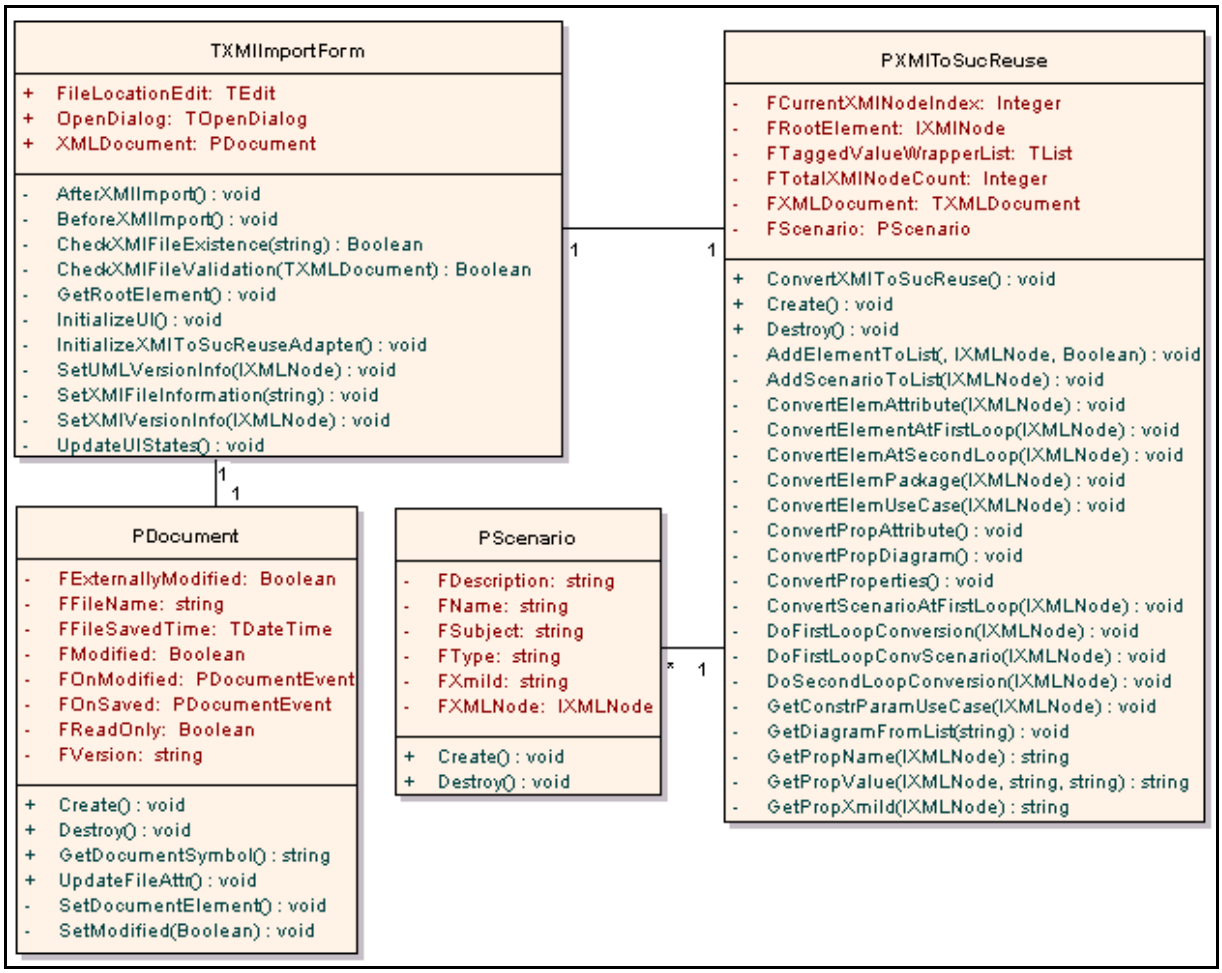
No Quadro 11 apresenta-se o diagrama de classe da manipulação do metadados do arquivo XMI a ser importado, baseado no código fonte do software StarUML (PLASTIC, 2006).

A classe *PDiagram* contém os atributos e métodos comuns a todos os diagramas, sendo a classe base da classe *PUMLDiagram* que possui os métodos e atributos comuns a todos os diagramas da UML, já a classe *PUMLUseCaseDiagram* possui os atributos e métodos comuns aos diagramas de casos de uso.



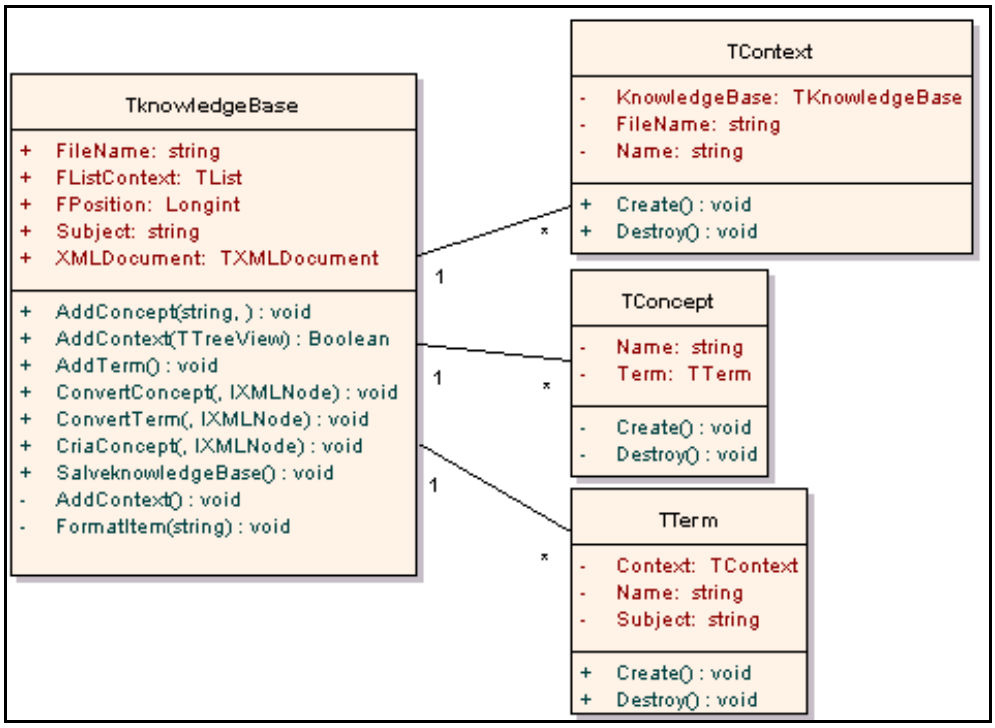
Quadro 11 – Diagrama de classes: Manipulação do metadados

A seguir o Quadro 12 representa o diagrama de classe da importação do arquivo XMI. A classe *TXMIImportForm* é a classe de interface da importação do arquivo, onde estão os métodos de manipulação do início da importação do arquivo, sendo também responsável pela verificação da versão do arquivo e integridade do arquivo. A classe *PXMIToSucReuse* importa todos os elementos do diagrama de casos de uso presente no arquivo XMI. Como cenário não é um elemento da UML, foi criado a classe *PScenario* específica para este elemento dos casos de uso.



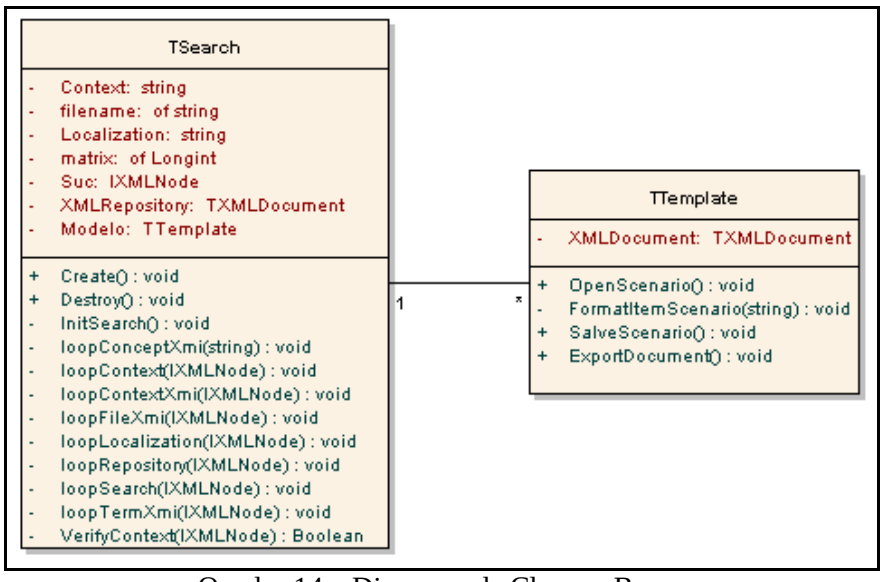
Quadro 12 – Diagrama de classes: Importação do arquivo XMI

A seguir o Quadro 13 representa o diagrama de classe da base de conhecimento. A classe *TKnowledgeBase* está associada com as classes *TContext, TTerm* e *TConcept.* Os métodos *AddContext, AddTerm* e *AddConcept* são responsáveis por associar um termo a um ou mais conceitos e que estará atribuído a um contexto.



Quadro 13 – Diagrama de classes: Base de conhecimento

A seguir o Quadro 14 representa o diagrama de classe da busca. A classe *TSearch* é responsável por realizar a busca em todos os modelos de casos de uso armazenados no repositório da ferramenta. O método *InitSearch* inicializa a busca, utilizando o método *loopSearch* para alimentar uma matriz com os nomes dos arquivos que foram encontrados e a relevância do arquivo que contém o modelo de casos de uso, dentro do contexto da busca, está relevância é calcula pela quantidade de ocorrência do termo pesquisado na busca.



Quadro 14 – Diagrama de Classes: Busca

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentados os aspectos sobre a implementação do protótipo e as ferramentas utilizadas para a construção.

3.3.1 Técnicas e Ferramentas Utilizadas

Para implementação do protótipo foi utilizado o ambiente de desenvolvimento Borland Delphi 7. A linguagem utilizada pelo ambiente é Object Pascal. No Quadro 15 é apresentado um trecho do código responsável pela criação do repositório dos modelos de casos de uso.

```
//Obriga informar uma localização para o repositório
ConsistLocalizationRepository;
//Obriga informar pelo menos um contexto para o repositório
ConsistRepositoryContext;
//Cria objeto Repositório
Repository := TRepository.Create;
Repository.FileName := edFile.Text;
Repository.NewRepository(Main.FileCfgRepository.XMLDocument);

RepositoryContext := TRepositoryContext.Create;
//Criar o Array de Context
CfgArrayRepositoryContext;
//Criar contextos do repositório no xml
for i:= 1 to tvContext.Items.Count-1 do
     begin
     RepositoryContext.NewRepositoryContext(Main.FileCfgRepository.XMLDocument,
                                            tvContext.Items[i].Text);
     Repository.AddRepository(RepositoryContext.XMINode);
     end;
Main.FileCfgRepository.AddFileCfgRepository(Repository.XMINode);
Main.FileCfgRepository.SaveFileCfgRepository(Main.FileCfgRepository.XMLDocument);

//Salvar no registro do windows
Main.FileCfgRepository.WriteRegisty('Repository',Repository.FileName);
Main.FileCfgRepository.WriteRegisty('Context',tvContext.Items[1].Text);
//Seta o repositório e contexto padrão da ferramenta
Main.FileCfgRepository.DefautRepository := Repository.FileName;
Main.FileCfgRepository.DefautRepositoryContext := tvContext.Items[1].Text;
```

Quadro 15 – Inicializa sistema

No Quadro 16 o método ConvertXMIToSucReuse chama as rotinas do mecanismo de leitura do arquivo XMI para a ferramenta SucReuse.

```
procedure PXMIToSucReuse.ConvertXMIToSucReuse(RootElem: PUMLPackage);
var
  Content: IXMLNode;
  Extensions: IXMLNode;
begin
  if (XMLDocument <> nil) and
     (RootElem <> nil) then
  begin
    FRootElement := RootElem;
    FProject := Project;
    XMLUtil.XMLDocument := XMLDocument;
    //Pega o Nodo Context do arquivo para importar os casos de uso
    Content := XMLUtil.FindFirstNode(nil, NODE_XMI_CONTENT);
    if Content <> nil then begin
      PrepareConversion(Content);
      DoFirstLoopConversion(Content);
      DoSecondLoopConversion(Content);
      FinalizeConversion;
    end;
    //Pega o Nodo Extensions do arquivo para importar os cenários
   Extensions  := XMLUtil.FindFirstNode(nil, NODE_XMI_EXTENSIONS);
    if Extensions <> nil then
      DoFirstLoopConversionScenario(Extensions);
  end;
end;
```

Quadro 16 – Método ConvertXMIToSucReuse

No Quadro 17 o método AddContext cria um contexto na base de conhecimento, que é salvo no caso de uso que está sendo modelado para o reuso na ferramenta.

```
procedure TFknowledgeBase.AddContext;
var
AContext : TContext;
begin
AContext := TContext.Create;
AContext.Name := main.FileCfgRepository.defautrepositorycontext;
AContext.FileName := main.FileCfgRepository.defautrepository+'\'+
main.FileCfgRepository.defautrepositorycontext+'\'+ExtractFileName(FileName);
FListContext.Add(AContext);
end;
```

Quadro 17 – Método AddContext

No Quadro 18 o método AddTerm adiciona um termo na base de conhecimento no contexto em questão.

```
procedure TFknowledgeBase.AddTerm(C: TContext);
var ATerm : TTerm;
i : Longint;
begin
ATerm := TTerm.Create;
ATerm.Name := FStringListTerm[FPosition-1];
ATerm.Context := C;
FListTerm.Add(ATerm);
for  i:= 0 to meConcept.Lines.Count - 1 do
     AddConcept(meConcept.Lines[i],ATerm);
end;
```

Quadro 18 – Método AddTerm

No Quadro 19 o método AddConcept adiciona um conceito na base de conhecimento para o termo em questão.

```
procedure TFknowledgeBase.AddConcept(S: String; T: TTerm);
var AConcept : TConcept;
begin
AConcept := TConcept.Create;
AConcept.Name := S;
AConcept.Term := T;
FListConcept.Add(AConcept);
end;
```

Quadro 19 – Método AddConcept

No Quadro 20 mostra a busca para encontrar os arquivos de maior relevância em relação a busca do usuário.

```
Search := TSearch.Create;
try
     SetLength(Search.Matrix, clConcept.Count+1);
     Search.InitSearch();
     for  j := 0 to Length(Search.Matrix[0]) - 1 do
          begin
          AuxVar := 0;
          for  i := Low(Search.Matrix) to High(Search.Matrix) do
               AuxVar := AuxVar + Search.Matrix[i,j];
          Search.Matrix[Length(Search.Matrix)-1,j] := AuxVar;
          end;
     for i:= Low(Search.filename) to High(Search.FileName) do
          ValueListEditor.InsertRow(Search.filename[i],
          IntToStr(Search.Matrix[Length(Search.Matrix)-1,i]),true);
finally
     Search.Free;
end;
```

Quadro 20 – Busca

3.3.2 Operacionalidade da implementação

A Figura 03 mostra o diagrama de atividades, apresentado de forma macro o funcionamento da ferramenta.

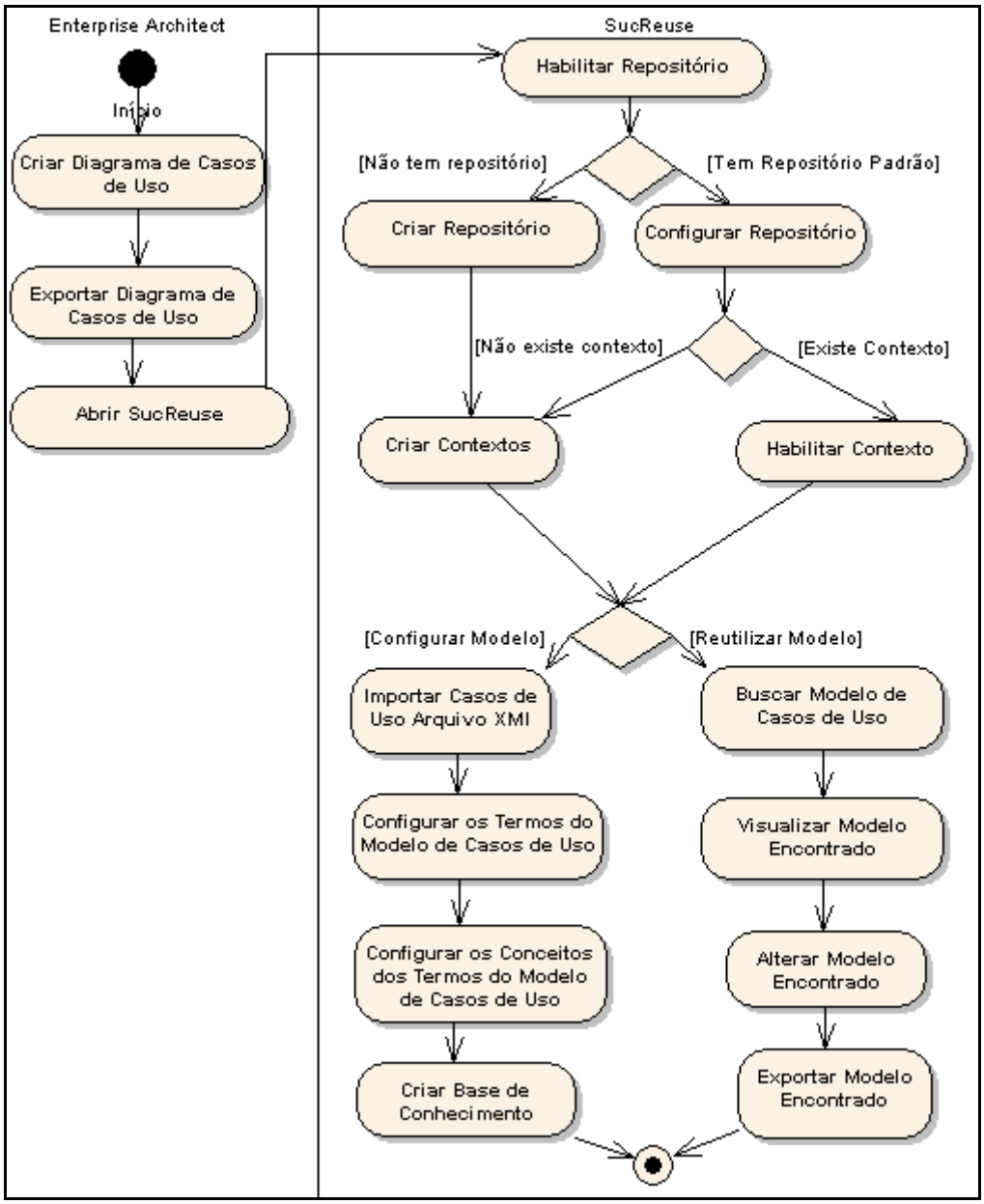


Figura 03 – Diagrama de atividade: funcionamento da ferramenta

Quando a ferramenta é executada pela primeira vez, a tela apresentada na Figura 04 é exibida, é informando ao engenheiro de software que será criado/editado o arquivo SucRepository.xml, que possui a definição dos repositórios conforme mostra a Quadro 21.

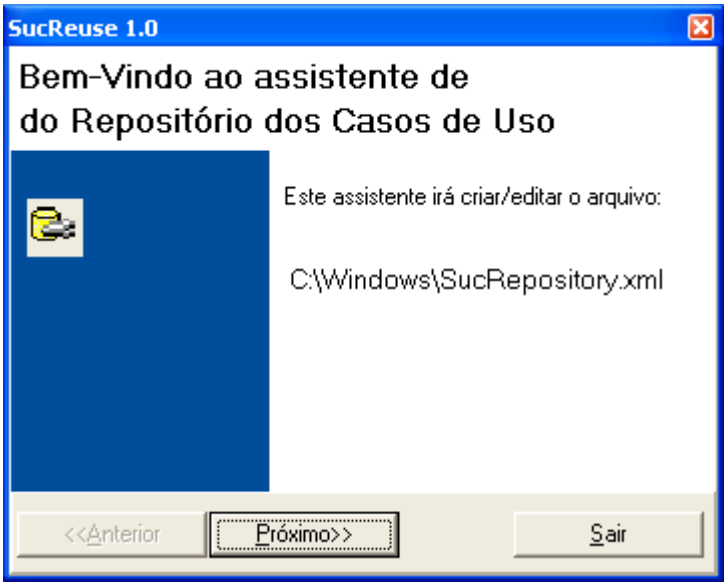


Figura 04 – Tela de abertura do repositório

```
<XMI xmi.version = "1.1" xmlns:UML="href://org.omg/UML/1.3" timestamp = "Tue May 02 16:39:37
      <SUC>
            <SUC.Repository filename="C:\Modelo de Projeto">
                  <SUC.RepositoryContext name="Cadastro Geral"/>
                        <SUC.Localization directory="C:\Modelo de Projeto\Cadastro Geral">
                              <SUC.File name="uscdgrupoempresa.xmi"/>
                              <SUC.File name="uscdbanco.xmi"/>
                        </SUC.Localization >
                  </SUC.RepositoryContext>
                  <SUC.RepositoryContext name="Movimento Financeiro"/>
                        <SUC.Localization directory="C:\Modelo de Projeto\Movimento Financeiro">
                              <SUC.File name="usmfAplicacaoFinanceira.xmi"/>
                              <SUC.File name="usmfBaixaTituloFinanceiro.xmi"/>
                        </SUC.Localization >
                  </SUC.RepositoryContext>
            </SUC.Repository>
      </SUC>
</XMI>
```

Quadro 21 – Exemplo do Arquivo SucRepository.xml

Na Figura 05 apresenta-se a possibilidade de criar um novo repositório, ou configurar um repositório existente, que é um diretório onde todos os modelos gerados pela ferramenta devem ser salvos.

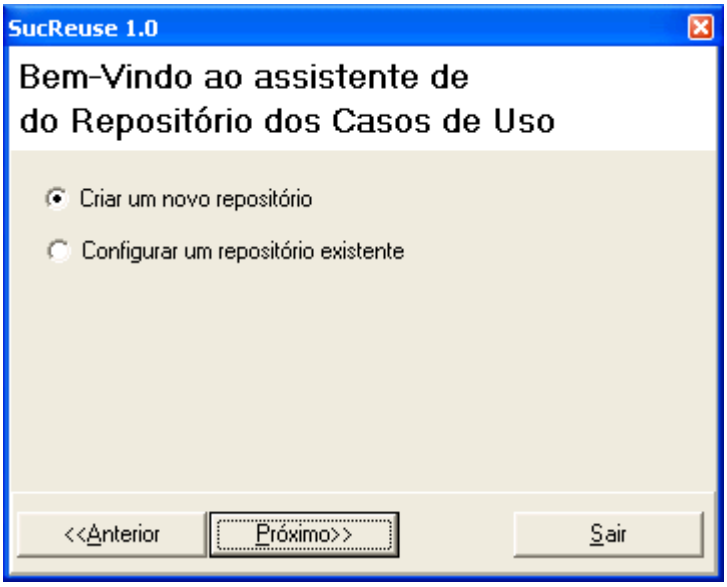


Figura 05 – Tela de criação e configuração do repositório

Ao escolher por criar um novo repositório, o próximo passo, Figura 06, é definir a localização do repositório e todos os contextos possíveis deste repositório. Posteriormente poderá ser criados novos contextos para este repositório, através da opção configurar um repositório existente.

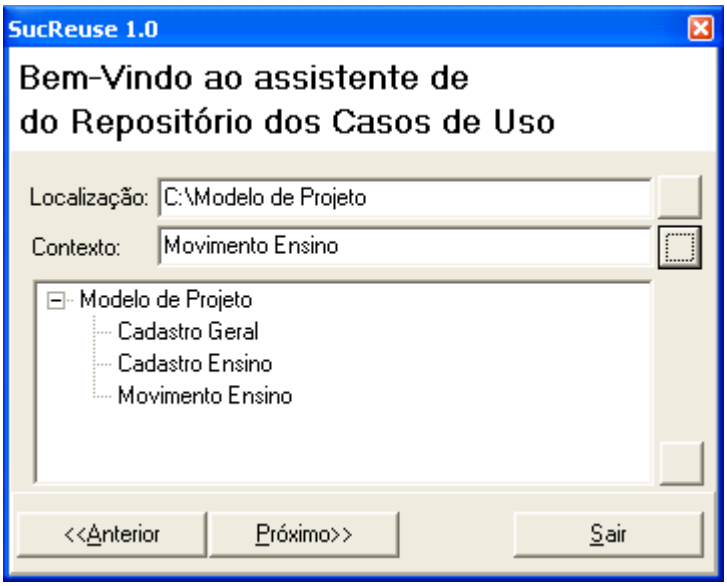


Figura 06 – Tela de criação do repositório e contextos

O processo de reuso se baseia nos casos de uso que estão armazenados neste repositório, portanto, o engenheiro de software especialista no domínio que se deseja desenvolver os modelos de casos de uso para serem reutilizados, deve criar repositórios para cada tipo de especificação de casos de uso. Por exemplo: descrição inicial, descrição base,

descrição elaborada. Desta forma tem-se o grau de abstração desejada no reuso.

Na Figura 07 apresenta-se a mensagem que o repositório foi criado com sucesso. O primeiro repositório criado e o primeiro contexto criado serão utilizados como o repositório padrão da ferramenta e o contexto padrão da ferramenta, sendo possível alterá-los durante o uso da mesma.

Figura 07 – Repositório criado com sucesso

A Figura 08 apresenta a interface da ferramenta.

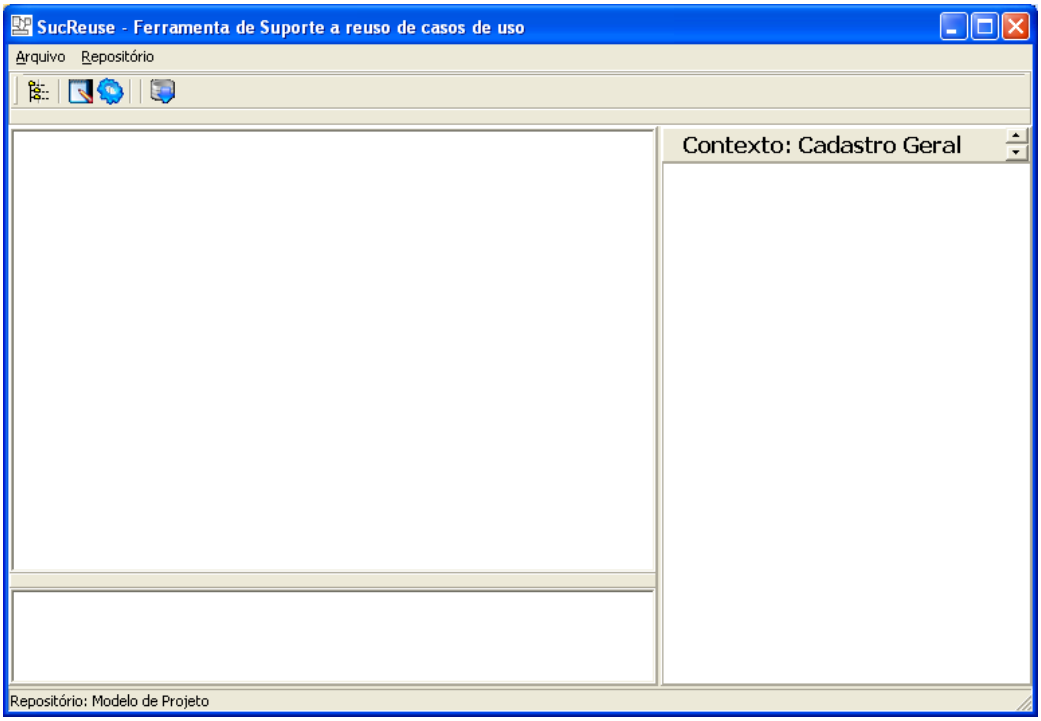


Figura 08 – Interface da Ferramenta

O ambiente é composto por um menu principal cujas opções são Arquivo e Repositório. Abaixo do menu principal existe uma barra de ferramentas para acesso dos comandos mais comuns. Ao lado direito está o contexto em que a ferramenta está configurada, dando a possibilidade de alterar para outro contexto. Na parte inferior informa o repositório que a ferramenta está usando.

O menu Arquivo, conforme a Figura 09 é subdividido nos seguintes itens:

Figura 09 – Menu arquivo

a) abrir XMI. Na Figura 10 apresenta-se a tela de seleção do arquivo XMI, que é chamada a partir deste item, ao clicar no botão “ok”, o caso de uso definido no arquivo será importado para a ferramenta SucReuse. Recuperando desta forma os casos de uso definido na ferramenta que o arquivo foi exportado. Nesta versão inicial da ferramenta o SucReuse suporta somente arquivos gerados pela ferramenta Enterprise Archictect.

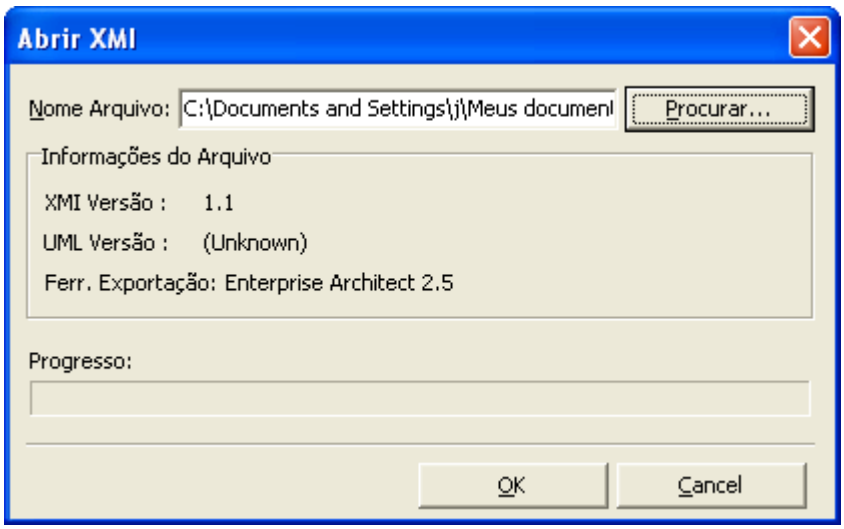


Figura 10 – Abrir XMI

b) salvar XMI. Exportar as alterações feitas pela ferramenta para o arquivo ser aberto em outra ferramenta de modelagem. Como exposto anteriormente, nesta versão somente para a ferramenta Enterprise Architect;

c) gerar modelo. Neste item será possível realizar buscas inteligentes através de uma semântica estabelecida nos modelos de casos de uso do repositório, após encontrar o modelo que mais atenda a necessidade do reutilizador, é possível gerar um modelo pronto para o reuso;

d) formatar cenário: Neste item é possível definir como os passos do cenário devem ser formatados, conforme Figura 11.

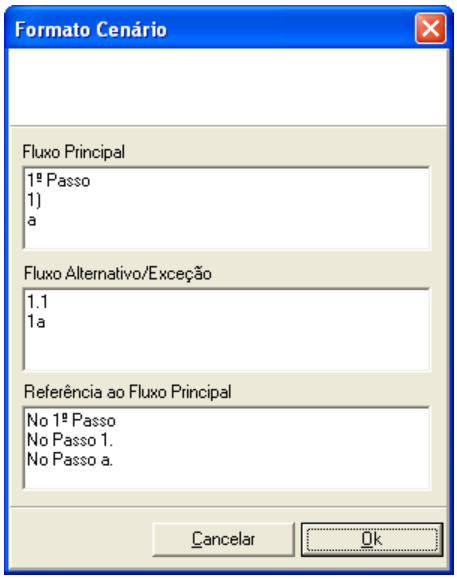


Figura 11 – Formata cenário

O menu Repositório, conforme a Figura 12 possui o seguinte item: configuração, nesta opção será permitido criar novos repositório para a ferramenta e novos contextos para os repositório.

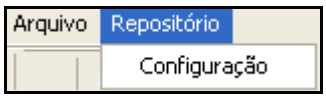


Figura 12 – Menu configuração

A Figura 13 mostra a descrição de cada botão da barra de ferramenta.

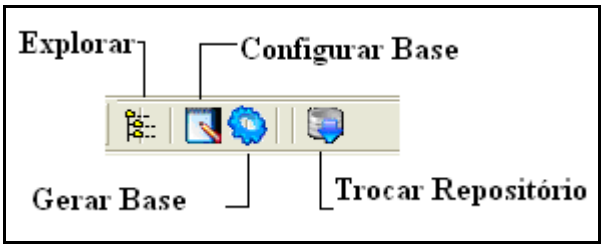


Figura 13 – Barra de Ferramentas

a) explorar: este botão possibilita visualizar a árvore do modelo do projeto gerado na ferramenta de modelagem Enterprise Archictect, mostrando todos os objetos do projeto e não apenas os casos de uso. Conforme mostra a Figura 14;

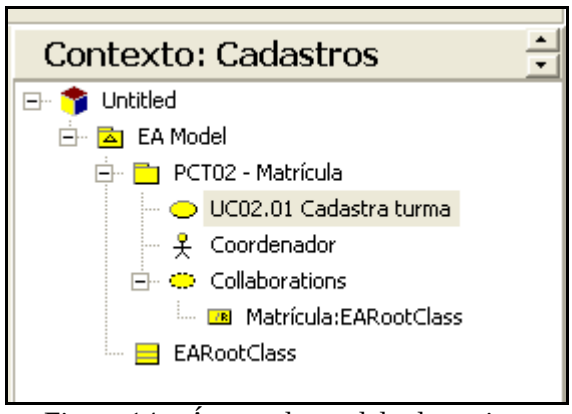


Figura 14 – Árvore do modelo do projeto

b) configurar base: para configurar a base de conhecimento dos casos de uso, deve possuir um contexto selecionado na ferramenta;

c) gerar base: ao clicar neste botão é gerado a base de conhecimento do caso de uso aberto na ferramenta;

d) trocar repositório: neste botão é possível selecionar outro repositório para armazenar os modelos de casos de uso gerados pela ferramenta.

Para iniciar o uso da ferramenta SucReuse o engenheiro de software deve gerar o diagrama de caso de uso na ferramenta Enterprise Architect conforme Figura 15 (exemplo extraído do material didático da professora Fabiane Barreto Vavassori Benitti), e exportar o caso de uso para um arquivo XMI.

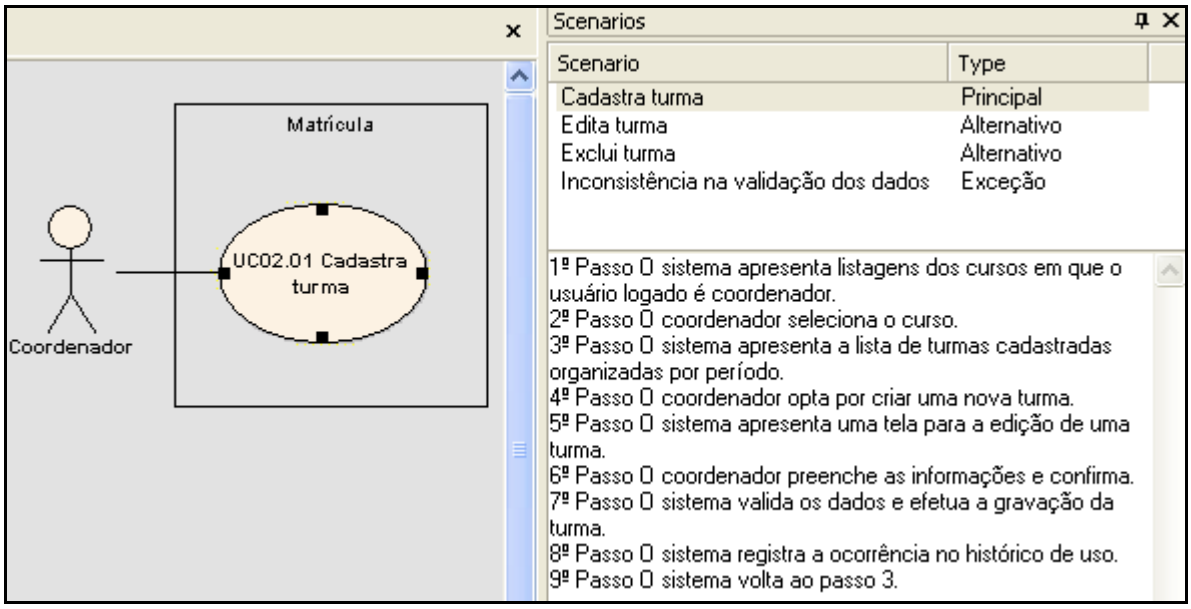


Figura 15 – Caso de uso na ferramenta Enterprise Architect

Após a exportação o arquivo com o caso de uso que foi gerado em outra ferramenta, deve ser importado na ferramenta SucReuse, na importação dos cenários dos casos de uso, o cenário será formatado conforme definido na opção Formata cenário (Figura 11).

A seguir deve ser analisado o cenário do caso de uso, e definir quais termos devem ser conceituados. A marcação dos termos é feita com abre chaves e fecha chaves. Na Figura 16 o engenheiro de software informa os conceitos de cada termo marcado no cenário do caso de uso. Exemplo dos termos marcados: curso, turma, coordenador, onde curso será conceituado como: palestra e seminário; turma: classe, sala, horário de palestra, horário de seminário, horário de aula; coordenador: palestrante, seminarista e professor.

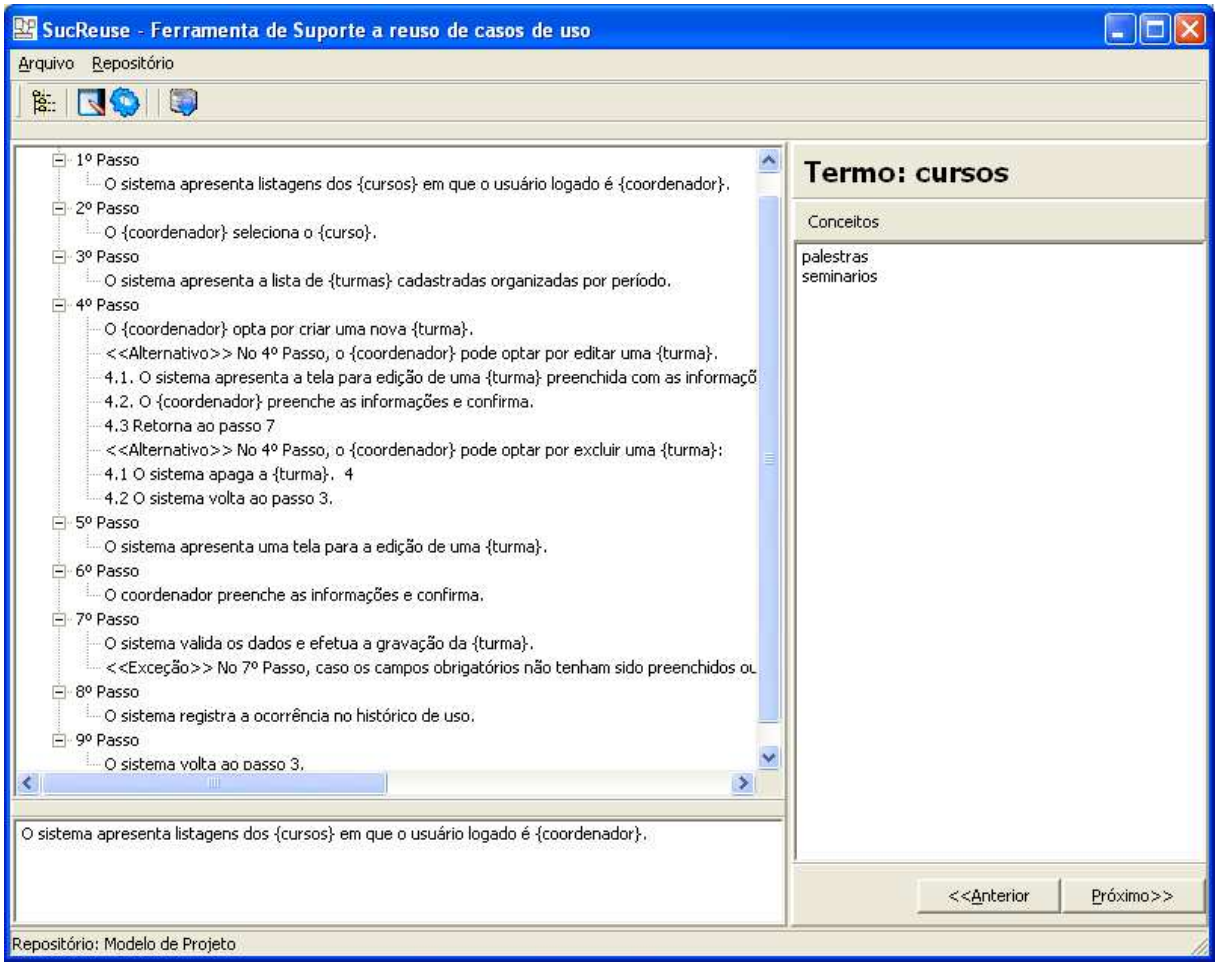


Figura 16 – Exemplo de marcação feita no caso de uso

Após a conceitualização de todos os termos do cenário deve ser gerada a base de conhecimento usando o botão “gerar base”. Os termos do modelo de caso de uso são salvos no arquivo XMI do caso de uso, conforme mostra o Quadro 22 e logo após a mensagem da Figura 17 é apresentada.

```
<XMI xmi.version="1.1" xmlns:UML="omg.org/UML1.3" timestamp="2006-05-13 14:31:03">
      <SUC.Knowledgebase>
            <SUC.Context name="Geral" filename="C:\Public\repository.xml">
                  <SUC.Term name="cursos" subject="">
                        <SUC.Concept description="palestras"/>
                        <SUC.Concept description="seminarios"/>
                  </SUC.Term>
                  <SUC.Term name="coordenador" subject="">
                        <SUC.Concept description="palestrante"/>
                        <SUC.Concept description="seminarista"/>
                  </SUC.Term>
                  <SUC.Term name="curso" subject="">
                        <SUC.Concept description="palestra"/>
                        <SUC.Concept description="seminario"/>
                  </SUC.Term>
                  <SUC.Term name="turmas" subject="">
                        <SUC.Concept description="plateias"/>
                        <SUC.Concept description="ouvintes"/>
                  </SUC.Term>
                  <SUC.Term name="turma" subject="">
                        <SUC.Concept description="plateia"/>
                        <SUC.Concept description="ouvinte"/>
                  </SUC.Term>
            </SUC.Context>
      </SUC.Knowledgebase>
</XMI>
```

Quadro 22 – Exemplo da Base de conhecimento criada no arquivo XMI

Figura 17 – Gerar base

O processo de reutilização dos casos de uso armazenados no repositório da ferramenta, inicia-se com a necessidade de encontrar um modelo de caso de uso que corresponda a necessidade do reutilizador. Portando o reutilizador a partir do escopo do problema ou o requisito funcional, deve realizar uma busca semântica, para encontrar um modelo que possua uma maior relevância.

Para realizar a busca deve ser informado os contextos que se deseja abranger conforme mostra a Figura 18.

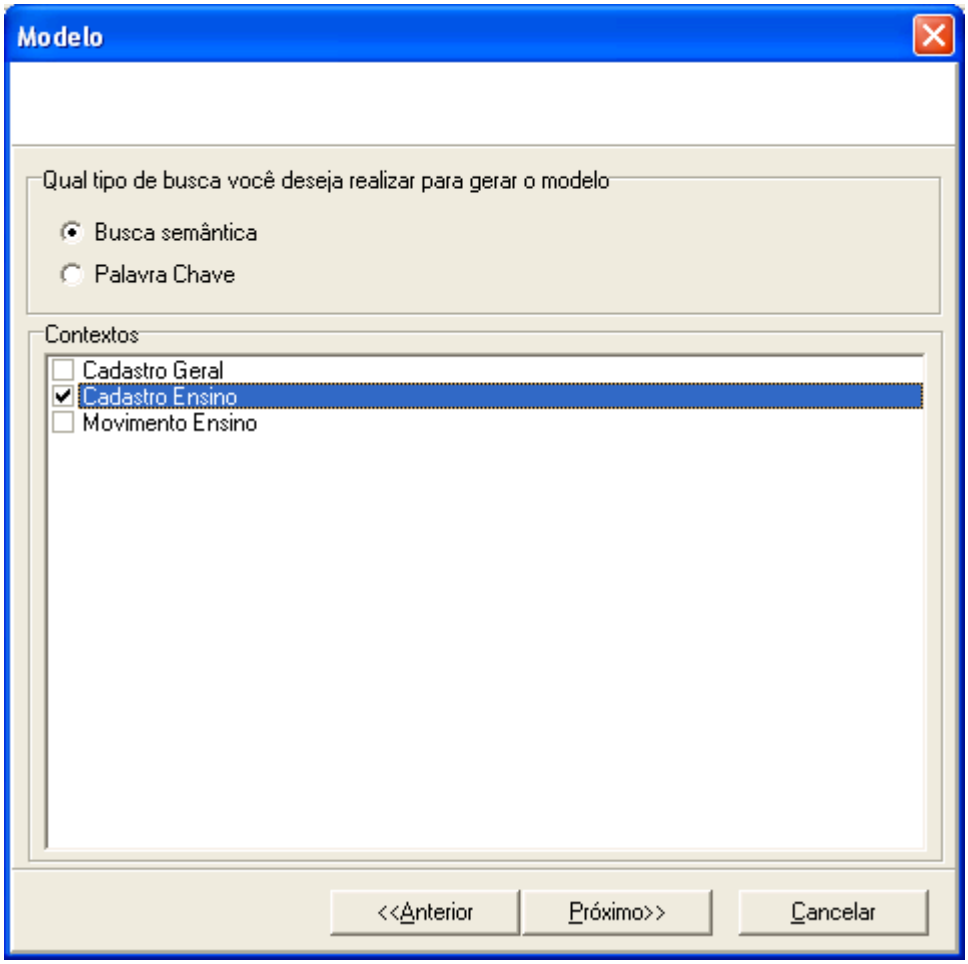


Figura 18 – Marcar contextos para busca de modelos de casos de uso

Após informado os contextos, o usuário deverá ir para o próximo passo. No passo apresentado na Figura 19, o usuário deve fazer uma descrição breve sobre o escopo do problema, para encontrar um modelo para o reuso. O sistema definirá cada palavra digitada como um conceito, sendo que, a partir destes conceitos, o mecanismo de busca encontrará os termos que foram marcados pelo engenheiro de software na elaboração do modelo de caso de uso armazenado no repositório.

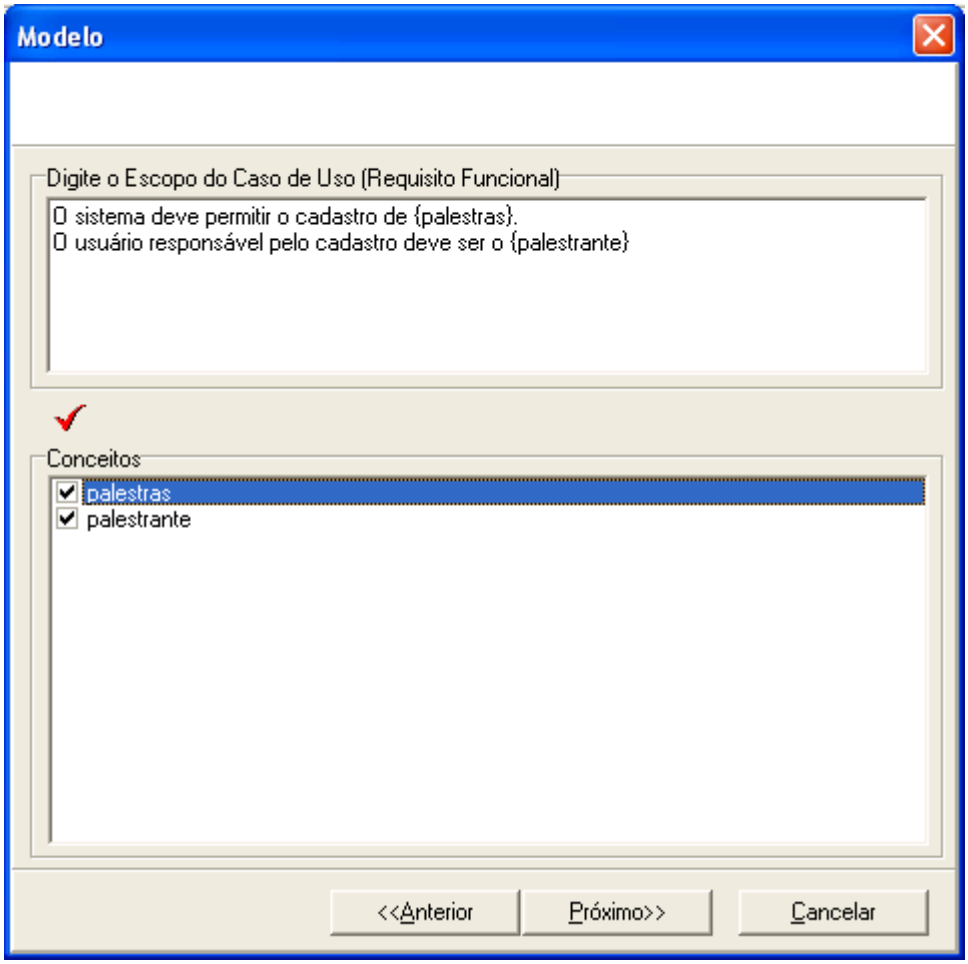


Figura 19 - Marcar conceitos para busca de modelos de casos de uso

No passo final o sistema irá realizar inferências na base de conhecimento de cada modelo de caso de uso (Figura 20), para encontrar um modelo com maior relevância.

Ao término da inferência o sistema irá apresentar a lista de casos de uso, em ordem de relevância. Após o reutilizador escolher o modelo que melhor represente o seu problema, o sistema irá substituir os termos marcados no modelo com os conceitos digitados na busca.

Após este processo, o reutilizador pode optar por ajustar os cenários conforme sua necessidade e exportar o modelo para uma ferramenta de modelagem utilizada por ele.

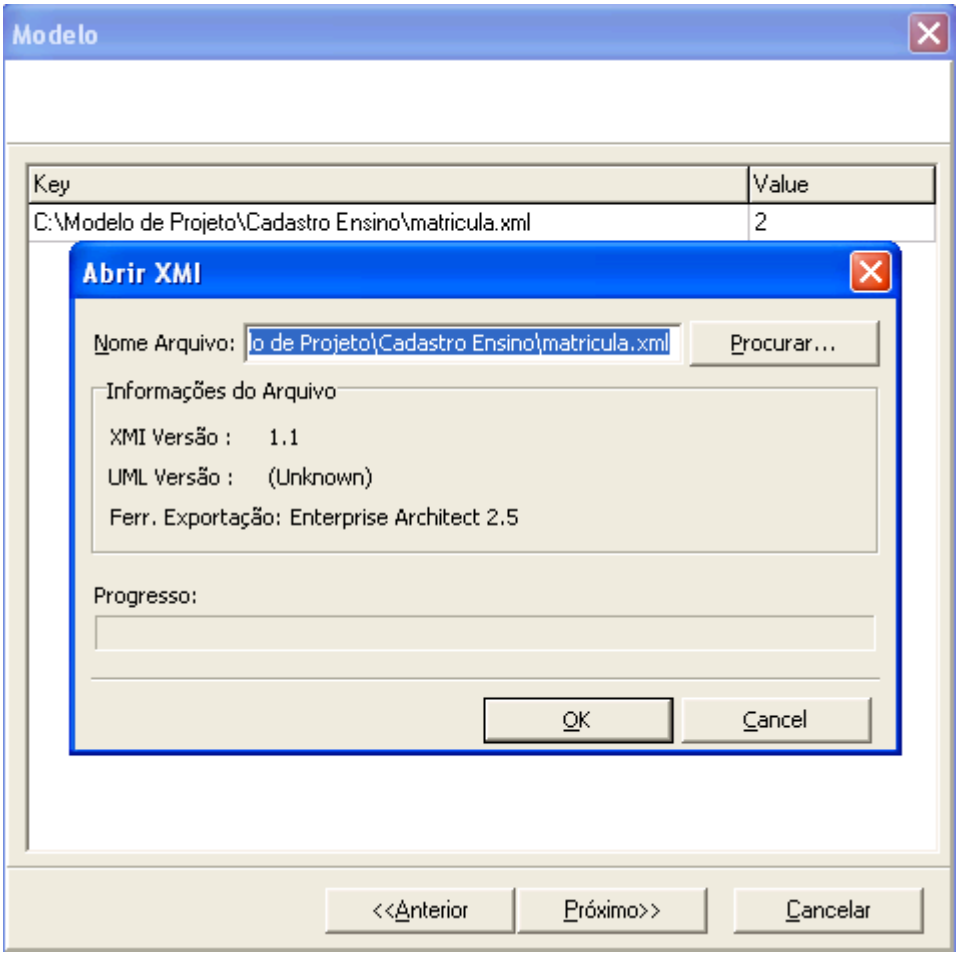


Figura 20 – Abrir modelo com maior relevância encontrado na busca

3.3.3 Estudo de Caso

O trabalho abordou a reutilização de casos de uso como forma de promover o reuso na fase de requisitos. Para verificar a funcionalidade da ferramenta, utilizando o raciocínio por analogia para contextualizar os modelos de casos de uso, foi identificado dentro de um sistema as seguintes funcionalidades comuns entre si, conforme Quadro 23.

Grupo de Empresas	↔ Grupo de Filiais
Títulos Financeiros a Pagar	↔ Títulos Financeiros a Receber
Pagamento Títulos Financeiros a Pagar	↔ Pagamento Títulos Financeiros a Receber
Empréstimos Financeiro	↔ Aplicações Financeiras
Pagamento Eletrônico	↔ Cobrança Escritural
Pedido	↔ Ordem de Compra
Notas Fiscais de Saída	↔ Notas Fiscais de Entrada
Contrato de compra	↔ Contrato de venda
Cadastro de produtos	↔ Cadastro de Serviços
Banco	↔ Portador

Quadro 23 - Analogias

Após este levantamento foram criados os contextos: cadastro geral, movimento financeiro e movimento comercial. Foram elaborados os seguintes modelos de casos de uso: grupo de empresas, títulos financeiros a pagar, pagamento títulos financeiros a pagar, empréstimos financeiros, pagamento eletrônico, pedido, notas fiscais de saída, contrato de compra, cadastro de produtos, banco.

Dentro do contexto “cadastro geral” foi inserido o caso de uso: grupo de empresas. No contexto “movimento financeiro” foi inserido os casos de uso: títulos financeiros a pagar, pagamento títulos financeiros a pagar, empréstimos financeiros, pagamento eletrônico e banco. Por último no contexto “movimento comercial” foi inserido os casos de uso: pedido, notas fiscais de saída, contrato de compra e cadastro de produtos. Após este processo foi feito um levantamento dos termos variáveis nos cenários dos casos de uso, onde estes termos foram marcados para serem conceituados, fazendo desta forma a ligação semântica necessária para realizar as buscas nos modelos incluídos no repositório da ferramenta.

O usuário da ferramenta que deseja elaborar um caso de uso sobre ordem de compra,

por exemplo, pode digitar um escopo ou algumas palavras em relação ao processo de uma ordem de compra. Como na elaboração do modelo de casos de uso “ordem de compra” foi encontrada analogia com pedido, então foi criada uma ligação semântica de pedido com ordem de compra, portando quando o usuário realizar a busca, este caso de uso será sugerido como um modelo para ser reusado para construir o modelo de caso de uso “ordem de compra”.

3.4 RESULTADOS E DISCUSSÃO

No estudo de caso mostrado na seção anterior a analogia se mostrou eficaz para identificar comportamentos comuns. O uso da ferramenta para criar uma base de conhecimento sobre o caso de uso para posteriormente serem reutilizados por problemas análogos pode ser uma forma bastante útil de encontrar casos de uso prontos para o reuso. Considerando uma grande quantidade de modelos de casos de uso análogos em um repositório, onde todos foram armazenados com ligação semântica, a possibilidade de o reutilizador encontrar um modelo bem próximo da sua necessidade é muito grande. Pois a técnica de busca usada considera a texto que mais se aproxima a necessidade do usuário.

O objetivo foi demonstrar a viabilidade de reutilizar casos de uso por analogia. O trabalho mostrou todos os passos necessários para a reutilização de casos de uso, utilizando sistemas de mesmo domínio a ferramenta demonstra uma facilidade na derivação dos casos de uso, comprovando assim a viabilidade de se reutilizar casos de uso por analogia.

4 CONCLUSÕES

O padrão XMI permitiu a utilização de casos de uso elaborados na ferramenta de modelagem Enterprise Architect para serem importados para a ferramenta desenvolvida neste trabalho. Desta forma, tornou-se possível a integração da ferramenta SucReuse com a ferramenta de modelagem Enterprise Architect.

O aprendizado na manipulação dos modelos UML, nos metadados em formato XMI pode-se dizer que foi muito enriquecedor, pois permitiu uma visão prática da utilização do XML na definição de um padrão.

Pode-se dizer que os objetivos definidos foram alcançados. O poder da analogia é seu potencial de absorver conhecimento de um domínio e aplicá-lo a um outro domínio foi de grande utilidade neste trabalho, pois desta forma foi possível identificar comportamentos comuns em problemas já resolvidos, visando desta forma o reuso.

O repositório integrado a ferramenta permitiu a separação dos modelos em casos de uso por contextos. Com esta classificação que é a ligação dos termos com seus conceitos, foi possível criar uma base de conhecimento nos casos de uso. Esta ligação semântica serviu de base para elaborar um mecanismo de busca semântica nos arquivos XMI.

4.1 EXTENSÕES

Para a continuidade ao trabalho, sugere-se as seguintes extensões:

a) desenvolver uma funcionalidade para visualizar casos de uso graficamente;
b) criar um repositório segundo o modelo Cliente/Servidor;
c) permitir a integração com todas as ferramentas de modelagem que utilizam o padrão XMI para exportar modelos;

d) permitir uma maior flexibilidade na marcação dos termos dos cenários e na alteração do texto dos cenários.

REFERÊNCIAS BIBLIOGRÁFICAS

BLASCHEK, Roberto J. **Gerência de requisitos:** o principal problema dos projetos de software. Rio de Janeiro, 2002. Disponível em: <http://www.bfpug.com.br/islig-rio/download.htm>. Acesso em: 25 maio 2006.

BASILI, V.R., CALDIERA G. Identifying and qualifying reusable software components. **IEEE Transactions on Software Engineering,** Piscataway, v. 24, n. 2, p. 61- 70, Fev. 1991.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML.** Rio de Janeiro: Campus, 2002.

CERQUEIRA, Andréa. **Introdução a metadados**. 2004. Disponível em: <http://www.linhadecodigo.com.br>. Acesso em: 25 maio 2006.

COCKBURN, Alistair. **Escrevendo casos de uso eficazes**. Tradução Roberto Vedoato. Revisão Marcelo S. Pimenta. Porto Alegre: Artmed, 2005.

CMMI. **CMMI Product Team**. 2002. Disponível em: <http://www.sei.cmu.edu/cmmi/>. Acesso em: 27 abr. 2006.

HALL, P.A.V. Architecture-driven component reuse. In: INFORMATION AND SOFTWARE TECHNOLOGY, 41., 1999, Hallstrom **Proceedings…** Hallstrom: Butterworth-Heinemann, 1999. p. 963-968.

JACOBSON, Ivar et al. **Object oriented software engineering:** a use case driven approach. Wokingham: Addison Wesley, 1992.

JUSTINO, Gilvan. **Ferramenta de apoio ao processo de reutilização de especificação estruturada.** 1999. 82 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KEANE, M. Analogical asides on case-based reasoning. **Springer-Verlag**, Lecture Notes in Artificial Intelligence, Berlin, v. 837, p. 21-32, Set. 1994.

LUCRÉDIO, Daniel. **Extensão da ferramenta MVCASE com serviços remotos de armazenamento e busca de artefatos de software.** 2004. 75 f. Dissertação (Mestrado em Ciências da Computação) - Departamento de Computação, Universidade Federal De São Carlos, São Carlos.

MAIDEN, N. A.; SUTCLIFFE, A. G. Exploiting reusable specifications through analogy. **Communications of the ACM.** [s.l], v. 35, n. 4, p. 55-64, Abr. 1992.

MANNION, Mike et al. Reusing single system requirements from application family requirements. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 21., 1999, Los Angeles. **Proceedings...** Los Angeles: IEEE Computer Society Press, 1999. p. 453–462.

MCILROY, Douglas. Mass produced software components. In: REPORT ON A CONFERENCE SPONSORED BY THE NATO SCIENCE COMMITTEE, 2., 1969, Garmisch. **Proceedings…** Garmisch: Nato Science Committee, 1969. p. 138-150.

NAUR, Peter. Software engineering. In: REPORT ON A CONFERENCE SPONSORED BY THE NATO SCIENCE COMMITTEE, 1., 1968, Garmisch. **Proceedings…** Garmisch: Nato Science Committee, 1969. p. 231.

NAVEGA, Sergio C. Manipulação semântica de textos, os projetos Wordnet e LSA. In: INFOIMAGEM, 1., São Paulo. **Anais**... São Paulo: CENADEM, 2004. p. 1-12.

NEIGHBORS, J. M. An assessment of reuse technology after tem years. In: INTERNATIONAL CONFERENCE ON SOFTWARE REUSE, 3., 1994, Los Alamitos. **Proceedings...** Los Alamitos, California: IEEE Computer Society Press, 1994. p. 6-13.

OLIVEIRA, Toacy C. **Uma abordagem sistemática para a instanciação de frameworks orientados a objetos.** 2001. 177 f. Tese (Doutorado em Ciências da Computação) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

PLASTIC. **Plastic software**. 2006. Disponível em: <http://www.plasticsoftware.com/>. Acesso em: 25 maio 2006.

PIMENTA, Alexandre. **Especificação formal de uma ferramenta de reutilização de especificações de requisitos.** 1998. 122 f. Dissertação (Mestrado em Ciências da Computação) - Instituto de Informática, Universidade Federal do Rio Grande Do Sul, Porto Alegre.

RIBEIRO, Alexandro C.; FLORENTINI, Ana C. **O padrão XMI**: uma proposta para sua utilização em Bibliotecas. 2000. 36 f. Dissertação (Mestrado em Informática) - O Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

ROLLAND, Colette; SOUVEYET, Carine; ACHOUR, Ben C. Guiding goal modelling using scenarios. **IEEE Transactions on Software Engineering,** Piscataway, v. 24, n. 12, p. 1055–1071, Dec. 1998.

RUP. **Rational unified process**. 2003. Disponível em: < www.ibm.com/software/ awdtools/rup>. Acesso em: 27 abr. 2006.

RYAN, Mark. Defaults in specifications. **IEEE Transactions on Software Engineering,** Piscataway, v. 20, n. 12, p. 142–149, Jan. 1993.

SANTOS, Misael S. **Uma proposta para a integração de modelos de padrões de software com ferramentas de apoio ao desenvolvimento de sistemas.** 2004. 105 f. Dissertação (Mestrado em Ciências da Computação) – Departamento de Computação, Universidade Federal do Ceará, Fortaleza.

SILVA, Alberto M. R.; VIDEIRA, Carlos A. E. **UML, metodologias e ferramentas CASE.** Portugal: Inova, 2001.

SOMMERVILLE, Ian.; KOTONYA, Gerald. **Requirements engineering** (processes and Techniques). England: J Wiley & Sons, 1997.

SOUZA, Renato R.; ALVARENGA, Lídia. **A web semântica e suas contribuições para a ciência da informação**. Ciência da Informação, Brasília, v. 33, n. 10, 2004. Disponível em: <http://www.ibict.br/cienciadainformacao/>. Acesso em: 21 maio 2006.

SPARX SYSTEMS. **Enterprise architect advanced modeling with UML**. 2006. Disponível em: <http://www.sparxsystems.com/>. Acesso em: 25 maio 2006.

TITTEL, Ed. **Coleção Schaum XML**. Porto Alegre: Bookman, 2003.

VITHARANA, Padmal; ZAHEDE, Fatemeh M.; JAIN, Hemant. Knowledge-based repository scheme for storing and retrieving business components: a theoretical design and an empirical analysis. **IEEE Transactions on Software Engineering,** Piscataway, v. 29, n. 7, p. 649–664, Jul. 2003.

ZIRBES, Sérgio F. **A Reutilização de modelos de requisitos de sistemas por analogia:** experimentação e conclusões**.** 1995. 168 f. Tese (Doutorado em Ciências da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.