

**UCSAL – UNIVERSIDADE CATÓLICA DO SALVADOR
BACHARELADO EM INFORMÁTICA**

**Eder Modesto dos Santos
João Paulo Barroso de Aragão**

RASTREABILIDADE DE ARTEFATOS DE SOFTWARE

Salvador - BA
Junho/2009

Eder Modesto dos Santos
João Paulo Barroso de Aragão

RASTREABILIDADE DE ARTEFATOS DE SOFTWARE

Monografia apresentada por Eder Modesto e João Paulo Barroso de Aragão como requisito parcial para aprovação na disciplina Projeto Final.

Orientador: Prof. Antônio Cláudio Neiva.

Salvador - BA
Junho/2009

**UCSAL – UNIVERSIDADE CATÓLICA DO SALVADOR
BACHARELADO EM INFORMÁTICA**

RASTREABILIDADE DE ARTEFATOS DE SOFTWARE

Antônio Cláudio Neiva

Salvador - BA
Julho/2009

CERTIFICADO

Certifico que a presente memória RASTREABILIDADE DE ARTEFATOS DE SOFTWARE, foi realizada, sob a minha orientação, por **Eder Modesto dos Santos e João Paulo Barroso de Aragão**, constituindo o Projeto Final do curso de Bacharelado em Informática da Universidade Católica do Salvador – UCSal.

Salvador, 20 de Junho de 2009.

Orientador: _____
Antônio Cláudio Neiva

**Salvador – BA
Junho/2009**

RESUMO

Atualmente, as informações são de grande importância para se ter um maior controle dos processos no desenvolvimento de software, o que mostra a grande necessidade de ferramentas que possibilitem utilizar a rastreabilidade de software. Um grande desafio neste projeto é como possibilitar ao desenvolvedor criar links de rastreabilidade entre os requisitos e os artefatos (métodos). Este projeto de pesquisa tem como objetivo mostrar os conceitos da rastreabilidade de software e construir uma ferramenta que auxilie os desenvolvedores a utilizarem a rastreabilidade de software.

Palavras-Chaves: Rastreabilidade de artefatos, Requisitos, Engenharia de Software, Engenharia de requisitos.

ABSTRACT

Nowadays, information is of great importance to take greater control of processes in software development, which shows the great need to use tools that allow the traceability of software. A major challenge in this project is to enable the developer to create traceability links between requirements and artifacts (methods). This research project aims to demonstrate the concepts of traceability and build a software tool that assists developers to use the traceability of software.

Key Word: Traceability artifact, Requirements, Software Engineering, Requirements Engineering.

SUMÁRIO

1. INTRODUÇÃO	1
2. ENGENHARIA DE SOFTWARE	2
2.1 REQUISITO	3
2.1.1 CLASSIFICAÇÃO DOS REQUISITOS	6
2.1.1.1 REQUISITO FUNCIONAL	6
2.1.1.2 REQUISITO NÃO-FUNCIONAL	6
2.1.1.3 REQUISITO DE USUÁRIO	7
2.1.2 BONS REQUISITOS	7
2.2 ENGENHARIA DE REQUISITOS	9
2.3 ANÁLISE DE REQUISITOS ORIENTADO A OBJETOS	11
2.4 PROJETO DE SOFTWARE ORIENTADO A OBJETO (OOD)	12
3. RASTREABILIDADE	14
3.1 MÉTODOS DE RASTREABILIDADE	21
3.2 RASTREABILIDADE DE REQUISITOS E ARTEFATOS	21
4. PADRÃO XML	23
5. PROJETO	26
5.1 AMBIENTE	26
5.2 REQUISITOS	27
5.2 ANÁLISE DA FERRAMENTA	27
5.2.1 DIAGRAMA DE CASO DE USO	28
5.2.2 ESPECIFICAÇÃO CASOS DE USO	28
5.2.3 DER	31
5.3 PROJETO TRACEFACTIN	31
5.3.1 ARQUITETURA	31
5.3.2 DIAGRAMA DE CLASSE	32
5.3.3 DIAGRAMA DE SEQUÊNCIA	33
5.4 USO DA FERRAMENTA	35
5.4.1 MENU	36
5.4.2 ASSOCIAR / DESASSOCIAR ARTEFATO	37
5.4.3 GERAR RELATÓRIO	38
5.4.4 CONSULTAR ARTEFATOS	39
5.4.5 IMPORTAÇÃO REQUISITOS	40

5.5 AVALIAÇÃO DA FERRAMENTA.....	41
6. CONCLUSÃO.....	42
REFERÊNCIAS BIBLIOGRÁFICAS.....	43

LISTA DE FIGURAS

FIGURA 1 - CAMADAS DE ENGENHARIA DE SOFTWARE	2
FIGURA 2 – TIPOS DE REQUISITOS.....	7
FIGURA 3 - PROCESSO DE ENGENHARIA DE REQUISITOS.	10
FIGURA 4 - MAPA DOS REQUISITOS	10
FIGURA 5 - ANÁLISE ORIENTADA A OBJETO E MODELAGEM DE DADOS	11
FIGURA 6 - MODOS DE RASTREABILIDADE	15
FIGURA 7 - MODELO DE RASTREABILIDADE LOW- <i>END</i>	19
FIGURA 8 - MODELO DE RASTREABILIDADE HIGH- <i>END</i>	20
FIGURA 9 – ARQUIVO XML UTILIZANDO <i>SCHEMA XMLBASEDSRS</i>	25
FIGURA 10 – DIAGRAMA DE CASO DE USO	28
FIGURA 11 – DER	31
FIGURA 12 – ARQUITETURA DA FERRAMENTA.....	31
FIGURA 13 – DIAGRAMA DE CLASSE	32
FIGURA 14 – DIAGRAMA DE SEQUENCIA 1	33
FIGURA 15 – DIAGRAMA DE SEQUENCIA 2	33
FIGURA 16 – DIAGRAMA DE SEQUENCIA 3.....	34
FIGURA 17 – DIAGRAMA DE ATIVIDADE	35
FIGURA 18 – FORMULÁRIO MENU	36
FIGURA 19 – FORMULÁRIO DE ASSOCIAR / DESASSOCIAR ARTEFATO	37
FIGURA 20 – FORMULÁRIO GERAR RELATÓRIO.....	38
FIGURA 21 – FORMULÁRIO CONSULTAR ARTEFATOS	39
FIGURA 22 – FORMULÁRIO DE IMPLANTAÇÃO.....	40

LISTA DE TABELAS

TABELA 1 - FATORES DAS MUDANÇAS DOS REQUISITOS.....	5
TABELA 2 - BONS REQUISITOS.....	8
TABELA 3 – TAGS XMLBASEDSRS.....	24

LISTA DE ABREVIATURAS

CMMI - Capability Maturity Model Integration

IBM -International Business Machines

SWEBOK - Guide to the Software Engineering Body of Knowledge

UML - Unified Modeling Language

IEEE - Institution of Electrical Engineers

OOA - Object-oriented analysis

OOD - Object-oriented design

ISO - International Organization for Standardization

XML - eXtensible Markup Language

SGML - Standard Generalized Markup Language)

1. INTRODUÇÃO

Ao enfatizar a natureza do trabalho, busca-se compreender as características fundamentais e inerentes ao trabalho do analista de sistema e programador no que diz respeito à Rastreabilidade de Artefatos de Software.

Existem ferramentas no mercado como DOORS, IBM Requisite Pro e Caliber RM, que auxiliam os desenvolvedores a gerenciar o documento de requisitos, escrever casos de uso e utilizar a rastreabilidade de requisitos para melhorar a qualidade do software, mas ainda não são capazes de possibilitar ao desenvolvedor a rastreabilidade dos artefatos (métodos) com os requisitos, fazendo com que mudanças nos requisitos se tornem mais suscetíveis a erro.

Neste trabalho, é apresentada uma ferramenta que visa a fornecer suporte a rastreabilidade de artefatos de software, integrando o Visual Studio 2008 com o Requisite Pro, no qual é feita a especificação dos requisitos do sistema em um modelo bem definido para representação das informações da estrutura dos arquivos que compõem determinado projeto de software. Desta forma, com base em estudos e pesquisas, criamos um Add-In para Visual Studio 2008 chamado **TraceFactIn**. Pretende-se, aqui, mostrar que é possível prover a rastreabilidade de artefatos relacionando os métodos das classes geradas aos requisitos especificados.

Nesta monografia, pretendemos mostrar que, com esta funcionalidade implementada, os envolvidos poderão ter informações importantes a respeito do software como: maior visão do impacto que o software sofrerá caso haja uma mudança nos requisitos,

A continuação deste trabalho está dividido em 5 capítulos, começando pelo segundo capítulo 2, que apresenta o conceito de engenharia de software e requisitos. O capítulo 4 apresenta o conceito de rastreabilidade de software e os métodos de rastreabilidade, o capítulo 5 apresenta o desenvolvimento da ferramenta onde foi exposta toda a documentação do software, o capítulo 6 apresenta as considerações finais e uma proposta de trabalho futuro.

2. ENGENHARIA DE SOFTWARE

Segundo Friedrich Ludwig Bauer (1968) , "*Engenharia de Software é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em máquinas reais*".

A Engenharia de software deve oferecer meios (mecanismos) para o planejamento e o gerenciamento durante o processo de desenvolvimento de sistemas, ou seja, ela foca nos aspectos práticos da fabricação de um software. Seu surgimento ocorreu na década de 1960 no intuito de controlar a crise de software. Grande parte das empresas passaram a utilizar a engenharia de software no intuito de controlar as áreas de desenvolvimento.

Para Pressman (2003) a Engenharia de Software é uma tecnologia em camadas, sendo que a sua base é o foco da qualidade. A Figura 1 apresenta as camadas da Engenharia de Software. A evolução do sistema durante o processo de Engenharia de software esta ligada a práticas que resultam com as atividades da especificação, projeto, implementação, testes, e caracterizam-se pela ligação entre as camadas apresentadas na Figura 1, que são as camadas, ferramentas, métodos, processos e foco na qualidade.

De acordo com SWEBOK (2004) a engenharia de software esta particionada em requisitos de software, testes de softwares, ferramentas e métodos, qualidade, gerência de software, etc.



Figura 1 - Camadas de Engenharia de Software (Pressman, 2003)

(Guide to the Software Engineering Body of Knowledge, documento criado sob o patrocínio da IEEE com a finalidade de servir de referência sobre quais assuntos são considerados, de forma generalizada, pela comunidade como pertinentes à área).

Para atingir os objetivos: alto rendimento, redução de custos e eficiência, algumas ações e comportamentos durante o desenvolvimento do projeto são úteis:

- A idealização do projeto;
- Acompanhar os resultados;
- Utilizar ferramentas unificadas na organização;
- Monitorar as táticas de testes;
- Revisar os componentes produzidos pelo processo;
- Buscar concordância com os padrões de desenvolvimento de software.

Na busca da qualidade e eficácia no desenvolvimento de software tem-se a Engenharia de Requisitos que segundo Zave (1997) é uma das principais áreas de atividade da Engenharia de Software. Uma vez que a Engenharia de Requisitos é específica de atuação, apresenta avanços tecnológicos e meios próprios voltados para pesquisa em terminologia, métodos, linguagens e ferramentas que compõem o esforço de pôr em prática o campo de conhecimento gerado.

2.1 REQUISITO

Segundo Pressman (1995), o entendimento do que são os requisitos é uma parte importante para o sucesso do desenvolvimento de um software e caso os requisitos não sejam bem especificados, muito dificilmente o software atenderá todas as necessidades do cliente.

Os requisitos de um sistema são as descrições dos serviços que o sistema deverá prover e suas respectivas restrições. Além dessa definição citada acima, existem muitas outras na literatura.

Segundo a IEEE (*Institution of Electrical Engineers*), requisito de software é:

1. uma condição ou capacidade necessária para o usuário resolver um problema ou alcançar um objetivo;

2. uma condição ou capacidade que deve ser encontrada ou possuída por um sistema ou componente do sistema para satisfazer um contrato, padrão, especificação ou outro documento imposto formalmente;
3. uma representação documentada de uma condição ou capacidade como em (1) ou (2)

Segundo Shalloway e Trott (2001) existem algumas simples razões que fazem os requisitos serem mudados, como:

- A visão dos usuários de suas necessidades mudam com o resultado de suas conversas com os desenvolvedores, vendo assim mais possibilidades para o software.
- A visão dos desenvolvedores do problema dos usuários muda enquanto eles desenvolvem se tornando mais familiarizado com os usuários.
- O ambiente que o software esta sendo desenvolvido muda.

Segundo Kotonya e Sommerville (1996), existem algumas classificações das mudanças dos requisitos, veja Tabela 1.

Tabela 1 - Fatores das mudanças dos requisitos. (Kotonya; Sommerville, 1996)

FATOR DE MUDANÇA	DESCRIÇÃO
Erros, conflitos e inconsistências dos requisitos.	Enquanto um requisito é analisado e implementado, erros e inconsistências aparecem e devem ser corrigidos. Este problemas podem ser descobertos durante a análise de requisitos, validação ou depois na fase de desenvolvimento.
Evolução do conhecimento dos usuários.	Enquanto os requisitos são desenvolvidos, os usuários entendem melhor o que eles realmente querem do sistema.
Problemas técnicos, tempo e custo.	Problemas podem ser encontrados na implementação de um requisito. Pode ser muito custoso ou levar bastante tempo para implementar um certo requisito.
Mudança de prioridade dos clientes.	As prioridades dos cliente mudam enquanto o software esta sendo desenvolvido com uma mudança do ambiente de negócio ou o aparecimento de novos concorrentes, etc.
Mudanças de ambiente.	O ambiente em qual o sistema esta para ser instalado pode mudar então os requisitos tem que mudar para manter a compatibilidade.
Mudanças organizacionais.	A organização que pretende usar o sistema pode mudar sua estrutura e processos resultando em novos requisitos.

2.1.1 CLASSIFICAÇÃO DOS REQUISITOS

No processo de especificação dos requisitos é necessário que se saiba que tipo de requisito está sendo tratado, para que haja uma maior compreensão das necessidades do cliente.

2.1.1.1 REQUISITO FUNCIONAL

Requisitos funcionais são aqueles que definem a função do software. Uma função é definida informando suas entradas, como esta entrada deverá ser manipulada e as suas saídas. Normalmente estes requisitos manipulam e processam dados ou executam uma funcionalidade específica, gerando assim resultados precisos para o software. (Sommerville, 2003)

Para que estes requisitos possam ser identificados posteriormente, existem campos como: nome, número e sumário que também servem para explicar qual a funcionalidade do requisito e facilitar a rastreabilidade, mas para que essas informações sejam aproveitadas tem que ser extremamente legíveis. (Sommerville, 2003)

2.1.1.2 REQUISITO NÃO-FUNCIONAL

São os requisitos que informam as condições ou regras que o sistema deve atender ou qualidades específicas que o software deverá ter, é neste momento quando são postas as condições de funcionamento do sistema que podem estar relacionadas às propriedades emergentes do sistema como, por exemplo, tempo de resposta, espaço em disco e confiabilidade. (Sommerville, 2003)

2.1.1.3 REQUISITO DE USUÁRIO

Os requisitos de usuário descrevem funcionalidades que o usuário poderá executar no software, um dos modos de representar é através da utilização dos casos de uso. Segundo Wiegers (2003) um pré-requisito necessário para desenhar um software é conhecer o que os usuários pretendem fazer.

Veja a relação entre alguns tipos de requisitos na Figura 2.

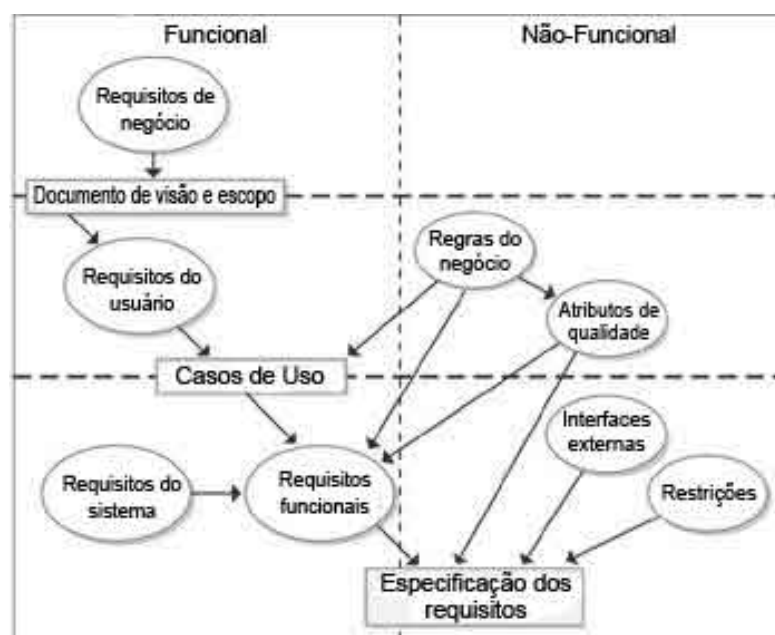


Figura 2 – Tipos de requisitos (Wiegers, 2003)

2.1.2 BONS REQUISITOS

Os requisitos precisam ser bem entendidos por todos que estão envolvidos no processo de análise de requisitos e para que isto se torne verdade, existem algumas características que normalmente são cumpridas, representadas na Tabela 2.

Tabela 2 - Bons Requisitos (Wiegers, 2003)

Coeso	O requisito deve atender única e exclusivamente uma funcionalidade.
Completo	O requisito deve explicar toda a informação necessária, sem deixar nenhuma informação perdida.
Consistente	O requisito não deve contradizer outro requisito e ser plenamente consistente com toda a documentação externa autorizada.
Correto	O requisito deve ter conhecimento de todo ou parte da necessidade do negócio previamente autorizada pelos stakeholders.
Corrente	O requisito não pode se tornar obsoleto com o passar do tempo.
Exteriormente observável	O requisito especifica uma característica do produto externamente observável pelo usuário. "Requisitos" que especificam uma arquitetura interna , modelam, implementam, ou testam decisões são restrições, e devem ser claramente explicadas na parte de restrições do documento de requisitos.
Possível	O requisito pode ser implementado dentro das restrições do projeto.
Não ambíguo	O requisito é claramente declarado sem nenhum recurso técnico ou específico. Ele expressa fatos e não opiniões subjetivas e tem somente uma interpretação. Declarações negativas e compostas são proibidas.
Obrigatório	O requisito representa uma característica cuja ausência resultará em uma deficiência que não se poderá melhorar.
Verificável	A implementação do requisito pode ser determinada através de um de quatro possíveis métodos: inspeção, análise, demonstração, ou teste.

2.2 ENGENHARIA DE REQUISITOS

A engenharia de requisitos é o processo que engloba todos os processos no desenvolvimento de um software. Neste processo é onde se inicia a elaboração do documento de requisitos e também sua manutenção ao longo do desenvolvimento. (Sommerville, 2003)

Seu objetivo é determinar as funções e restrições dos sistemas. Para que se obtenha sucesso é necessário executar atividades como:

1. Avaliação da viabilidade do sistema – com base em uma pequena descrição do sistema é realizado um estudo da necessidade e importância do seu desenvolvimento. Esta avaliação é registrada em um relatório que poderá conter informações sobre prazos, valores e requisitos;
2. Elicitação e Análise dos Requisitos – entrevistar os usuários para identificar as funções do sistema, verificar os tipos de usuários e compreender as suas necessidades. Existem alguns mecanismos para extrair do usuário as informações necessárias para a elaboração do documento de requisitos, como: Concepção, Levantamento, Elaboração, Negociação.
3. Especificação dos Requisitos – concentra –se na coleta e na organização de todos os requisitos que envolvem o projeto, é uma descrição da funcionalidade do sistema, das interfaces externas, desempenho e restrições.
4. Validação dos Requisitos – Diante da especificação analisa se os requisitos estão atendendo as necessidades do cliente. A validação é fundamental para evitar o retrabalho em etapas futuras.
5. Gerenciamento de requisitos – Gerenciar e controlar as modificações dos requisitos e a relação entre eles e o projeto.

Na Figura 3 tem-se uma representação do Processo de Engenharia de Requisitos.

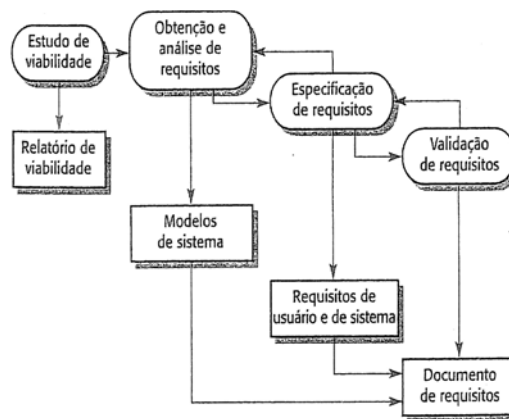


Figura 3 - Processo de Engenharia de Requisitos. (Sommerville, 2003).

Para compreender melhor sobre a Engenharia de Requisitos, é fundamental que no desenvolvimento do software, os usuários e os projetistas não só tenham o conhecimento aprofundado sobre suas características, ou seja, os requisitos do sistema.

Um requisito de software é uma descrição dos principais recursos de um produto de software, seu fluxo de informações, comportamento e atributos. Em suma, um requisito fornece uma estrutura básica para o desenvolvimento de um produto de software. O grau de compreensibilidade, precisão e rigor da descrição fornecida por um documento de requisitos de software tende ser diretamente proporcional ao grau de qualidade do produto resultante. (Peters, 2001)

Na figura 4 tem-se um mapa dos requisitos de sistema.

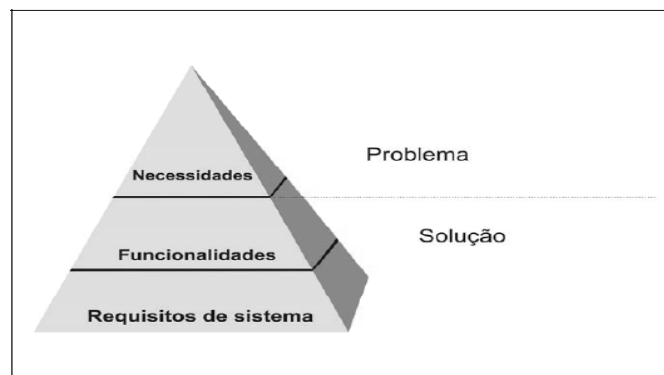


Figura 4 - Mapa dos requisitos. (Pressman, 2002)

2.3 ANÁLISE DE REQUISITOS ORIENTADO A OBJETOS

A análise de requisitos orientada a objetos (OOA) vem sendo aplicada cada vez mais nos dias atuais, pois com os conceitos da orientação a objetos esta metodologia torna-se mais poderosa, facilitando o estudo do domínio da aplicação. Para podermos falar de análise de requisitos orientado a objetos será necessário falarmos também sobre o conceito de orientação a objetos. Existem muitas respostas para o que é orientação a objeto, a seguir iremos mostrar um exemplo. (Pressman, 1995)

Para entender o ponto de vista orientado a objeto, consideremos um exemplo de objeto do mundo real, a mesma coisa em que você esta sentado agora, uma cadeira. Cadeira é um membro de uma classe muito maior de objetos, a qual denominamos mobiliário. Um conjunto de atributos genéricos pode ser associado a cada objeto da classe mobiliário. Por exemplo, todo mobiliário tem um custo, dimensões, peso, localização e cor, entre os muitos atributos possíveis. Esses atributos se aplicam, quer estejamos falando a respeito de uma mesa ou cadeira, de um sofá ou um armário. Uma vez que a cadeira é um membro da classe mobiliário, cadeira herda todos os atributos definidos para a classe. (Pressman, 2005)

Este exemplo citado acima é representado na figura 5.

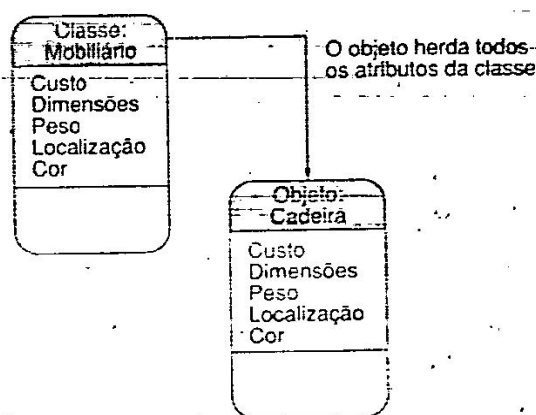


Figura 5 - Análise orientada a objeto e modelagem de dados. (Pressman, 1995)

A análise de requisitos orientada a objetos é uma maneira de possibilitar aos analistas fazerem a modelagem do sistema a ser desenvolvido utilizando classes para mapear os problemas do domínio da aplicação. Uma fase importante na

análise é a modelagem do domínio, pois é nela que são criados os diagramas para se ter noção da relação entre os objetos, dando assim uma maior visão ao analista do sistema. (Pressman, 2005)

2.4 PROJETO DE SOFTWARE ORIENTADO A OBJETO (OOD)

O projeto de software orientado a objetos se preocupa em como solucionar o problema e suas informações interessam ao programador e não ao cliente. Para solucionar o problema cria-se uma representação do domínio do problema do mundo real e leva-a a um domínio de solução que é o software. Diferente dos outros métodos, o projeto de software orientado a objeto, divide o domínio dos problemas em partes para que seja mais fácil de entender. (Pressman, 1995)

O que é projeto? É onde você se instala com um pé em dois mundos – o mundo da tecnologia e o mundo das pessoas e objetivos humanos - e você tenta juntar os dois.

O crítico romano de arquitetura Vitruvius adiantou a noção de que edifícios bem projetados eram aqueles que exibiam firmeza, comodidade e prazer. O mesmo poderia ser dito de bom software. Firmeza: um programa não deve ter quaisquer erros que inibam sua função. Comodidade: um programa deve ser adequado às finalidades as quais foi planejado. Prazer: a experiência de usar o programa deve ser agradável. (Engenharia de Software, Sexta Edição, Roger S. Pressman, 185p)

Geralmente os projetos se baseiam em padrões já existentes, onde se reusa elementos do projeto que foram aprovados com sucesso no passado. Muito utilizado também são as extensões de padrões que fornecem um esqueleto arquitetural para o projeto de software em um domínio de aplicação específico, chamado de Framework. (Pressman, 2006)

As funcionalidades que o software deverá atender, normalmente são expostas em formas de casos de uso. Um caso de uso é uma descrição narrativa de uma seqüência de eventos que ocorre quando um ator (agente externo) usa um sistema para realizar uma tarefa. (Jacobson, 1992)

Cada caso de uso descreve somente uma única funcionalidade do sistema. É comum a criação de diagramas de caso de uso usando a linguagem UML (*Unified Modeling Language*), que é uma linguagem visual de especificação, construção e

documentação de artefatos de software, (IBM Rational) mostrando assim ao desenvolvedor como o sistema se comportará quando forem feitas interações do mundo externo, visualizando-as em uma visão de alto nível.

Segundo Pressman (2006) existem sete critérios para um bom projeto de software:

1. Um projeto de software expor uma arquitetura que tem sido criada usando um padrão de projeto, ser composta de componentes que expõem boas características de projeto e possa que ser implementada de forma evolutiva, facilitando a implementação e teste.
2. Um projeto de software deve ser modular; o que é, o software deve ser logicamente particionado dentro dos elementos que desempenham funções específicas e sub funções.
3. Um projeto de software deve conter representações distintas de dados, arquitetura, interfaces e componentes.
4. Um projeto de software deve definir a estrutura de dados que são apropriadas para os objetos a ser implementados e que são retirados de padrões conhecidos.
5. Um projeto de software deve definir os componentes independente de suas características.
6. Um projeto de software deve exibir as interfaces que reduzem a complexidade das conexões entre os módulos e o ambiente externo.
7. Um projeto de software deve utilizar um método repetitivo impulsionado por informações obtidas durante a análise de requisitos do software.

3. RASTREABILIDADE

Segundo Edwards & Howell (1991), rastreabilidade é uma técnica que supri o relacionamento entre os requisitos, o projeto e a implementação final do sistema. Palmer (1997) declara que rastreabilidade da à assistência essencial no entendimento dos relacionamentos existentes entre requisitos de software, projeto e implementação.

Hamilton & Beeby (1991) vê a rastreabilidade como a habilidade de descobrir o histórico de cada funcionalidade do sistema. Greenspan & McGowan (1978) declaram que é a habilidade de permitir mudanças em todos os artefatos, requisitos, especificação e implementação.

Diferentes stakeholders como: gerente de projeto, analistas e usuários estão envolvidos no ciclo de desenvolvimento do software. A rastreabilidade precisa que esses stakeholders definam suas devidas diferenças em suas metas e prioridades, e muitos problemas de rastreabilidade surgem devido as suas diferenças e entendimento. (Ramesh; Edwards, 1993).

Na engenharia de requisitos, o maior desafio é construir os links entre os requisitos e os fontes. A rastreabilidade facilita a comunicação entre os envolvidos no projeto para aliviar esses problemas. Engenharia de requisitos pode ser facilitada capturando as informações necessárias para entender a evolução e verificação dos requisitos. (Fiksel 1992)

Durante o projeto, a rastreabilidade permite aos projetistas manterem o rastro do que acontece quando uma mudança requerida e implementada antes do software ser remodelado. (Edwards; Howell, 1991)

Rastreabilidade, segundo os padrões internacionais (ISO 8402), é definida como a habilidade de descrever a história, aplicação, processos e localização de um produto, a uma determinada organização, por meios de registros e identificação.

A evolução do sistema requer um melhor entendimento dos requisitos o qual pode ser conseguido traçando a rastreabilidade reversa de suas fontes. (Pinheiro; Goguen, 1996) A rastreabilidade prove a referencia cruzada dos itens da especificação de requisitos com os itens do projeto de software. (Dorfman; Thayer, 1990) Portanto, com a rastreabilidade completa, os custos são mais precisos e pode-se executar mudanças sem a dependência do engenheiro ou programador conhecerem as áreas afetadas por essas mudanças (Ramesh, 1999).

Na prática, rastreabilidade pode ser difícil de alcançar quando programadores estão envolvidos. Programadores são relutantes em manter a documentação atualizada, e a rastreabilidade é facilmente quebrada se artefatos (requisitos, itens do projeto, documentos e código) são alterados sem refazer sua relação com outros artefatos afetados pela mudança (Richardson; Green, 2004).

De um modo mais simples, rastrear é manter os registros necessários para identificar e informar os dados relativos à origem e ao destino de um produto. Manter a rastreabilidade seja de requisitos ou artefatos é um trabalho extenso e sem o auxílio de ferramentas especializadas esse trabalho provavelmente não poderá ser feito.

A rastreabilidade pode ser classificada de dois modos: pré-rastreabilidade e pós-rastreabilidade. No primeiro modo conecta-se os artefatos usados para criação dos requisitos com os requisitos criados, enquanto a segunda relaciona os requisitos com eles mesmos ou com outros artefatos. Mesmo a rastreabilidade sendo vantajosa raramente esta é estabelecida em ambientes de desenvolvimento de software. (Alan M. Davis, 1993)

Veja os dois modos básicos de rastreabilidade na figura 6.

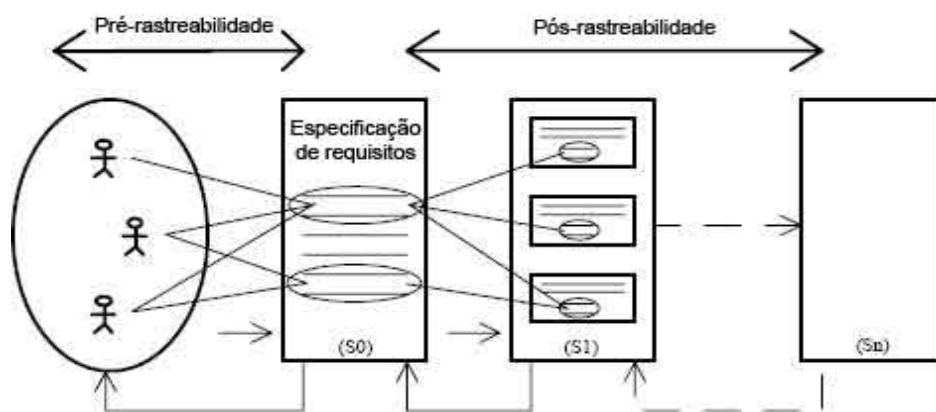


Figura 6 - Modos de rastreabilidade. (Anthony Finkelstein, O. G., 1994)

Segundo Spanoukadis (2005), existem oito grupos de links de rastreabilidade:

- Dependência: Neste tipo de relação um e1 depende do e2, se o e1 invoca o e2 ou se uma mudança no e2 reflete no e1. Ramesh & Jaker (2001) propõem o uso dos links de dependência entre os próprios requisitos e entre os requisitos e o projeto.
- Evolução: Este tipo de relação mostra a evolução dos artefatos de software. Neste caso o e1 evolui para um e2 e se e1 foi substituído pelo e2 durante o desenvolvimento, manutenção ou evolução do sistema.
- Satisfatibilidade: Neste tipo de relação um e1 satisfaz o e2, se o e1 atende as expectativas, necessidades e desejos do e2 ou se e1 esta satisfeito com as condições representadas pelo e2. Para Ramesh & Jaker (2001) as relações de satisfatibilidade são propostas como associações entre os requisitos de sistema e componentes e são utilizados para garantir que as exigências sejam satisfeitas por um sistema.
- Sobreposição: Neste tipo de relação um e1 sobrepõe um e2, se o e1 e o e2 referem-se à mesma funcionalidade ou ao mesmo domínio.
- Conflito: Este tipo de relação significa o conflito entre dois elementos, o e1 e e2. Para Ramesh & Jaker (2001) as relações de conflito são utilizadas para identificar os conflitos entre os requisitos, componentes, elementos e design, para definir questões relacionados a esses elementos, e para fornecer informações que possam ajudar a resolver os conflitos e as questões.
- *Rationale*: Este tipo de relação e usado para representar e manter o *rationale* atrás da criação e evolução dos elementos e decisões sobre o sistema em diferentes níveis de detalhe. Para Ramesh & Jaker (2001) as relações *rationale* são capturadas de acordo com o histórico das ações de como os elementos são criados.
- Contribuição: Este tipo de relação representa a associação entre os artefatos e os stakeholders que contribuíram para a geração dos requisitos.
- Generalização: Este tipo de relação é utilizada para identificar como elementos complexos do sistema podem ser quebrados em componentes, como elementos do sistema podem ser combinados para formar outro elemento e como um elemento pode ser refinado por outro elemento.

Os stakeholders têm prioridades diferentes durante o processo de desenvolvimento de software como: (Arkley, *et al*, 2006)

1. O gerente de projeto quer saber se todos os requisitos foram satisfeitos pelos componentes desenvolvidos.
2. O gerente de requisitos quer saber sobre a evolução dos requisitos, para poder ter mais informações sobre os impactos em outros requisitos quando houver alguma mudança.
3. Arquitetos de software querem manter uma rastreabilidade do que acontece quando for feita uma mudança, para poder ter mais opções para corrigir o sistema.
4. Os desenvolvedores podem ter mais precisão quando forem solicitados prazos para fazerem mudanças no sistema.

A manutenção da rastreabilidade é definida e obrigatória em alguns padrões de qualidade de software, como: MIL-STD-498, IEEE/EIA 12207 e ISO/IEC 12207. (Arkley, *et al*, 2006)

Em pesquisa recente (Arkley, *et al*, 2006) foram encontrados alguns problemas em manter a rastreabilidade atualizada, estes são:

1. Engenheiros sobrecarregados não querem documentar toda alteração que for feita;
2. Não é suficiente só documentar as justificativas, mas também as manter acessíveis;
3. Justificativas obsoletas são mais perigosas do que não criar justificativas.

Segundo Pohl (1996) os três principais problemas do uso das ligações de rastreabilidade são:

- O uso das informações de rastreabilidade depende da parte interessada e das atividades do processo de desenvolvimento de software, isto é, ligações não são usadas como são gravadas e, conseqüentemente, recuperações seletivas, de acordo com necessidades atuais tem de ser suportadas.
- Devido ao grande número de informações produzidas durante o processo, apenas ligações orientadas ao conteúdo são base para o uso apropriado, isto é,

o uso da informação gravada é quase impossível se as ligações de rastreabilidade não estão encapsuladas em seus contextos.

- As pessoas envolvidas na captura das ligações de rastreabilidade são frequentemente diferentes dos usuários da informação.

Uma das principais formas de resolver e manter a relação entre diferente artefatos é a habilidade de usar essas relações durante todo o ciclo de vida do artefato no software a fim de:

1. Resolver o impacto que potenciais mudanças feitas em alguma parte do sistema possa afetar outras partes do software;
2. Tomar decisões se tais mudanças devem ser desenvolvidas e com que prioridade

Um forma significativa para a geração de links de rastreabilidade é usá-los, para compreender os artefatos que tem referencia com o contexto que foram criados, ou em referencia a outros artefatos relacionados com eles. Esta rastreabilidade é importante nos casos em que as pessoas precisem ter acesso, entender e manter os artefatos que eles não contribuíram para a criação, um exemplo típico disso seria a manutenção do software. (Spanoukadis, 2005)

A compreensão do código-fonte é um dos principais objetivos da geração dos links de rastreabilidade entre os artefatos e os requisitos, testes e especificação do software (Marcus A. ; Maletic J.I., 2003). Impulsionado pela necessidade de entender o código-fonte surgiram novas abordagens que pode rastrear os artefatos a partir da semântica sobre a especificação do software.

Com base em estudo feito por (Ramesh, 1998) existem dois tipos de usuário que utilizam a rastreabilidade: *Low-End e High-End*.

Low-End: *Utilizam* os links rastreabilidade para localizar os componentes do sistema, informar os procedimentos de verificação, gerenciar as alterações do sistema em desenvolvimento e na manutenção do ciclo de vida do software, mas eles não enxergam a rastreabilidade como uma importante tarefa no seu processo de desenvolvimento e não utilizam nenhuma forma da “metodologia formal” em sua prática da rastreabilidade. Normalmente estes usuários tem algumas características como:

- Complexidade dos sistemas tem aproximadamente 1000 requisitos
- Experiência de zero a dois anos em rastreabilidade
- A definição de rastreabilidade é o documento de transformação dos requisitos para o projeto.
- As principais aplicações são controle de versão, localização dos requisitos e verificação de conformidade.

Veja na figura 7 o modelo de rastreabilidade *Low-End*.

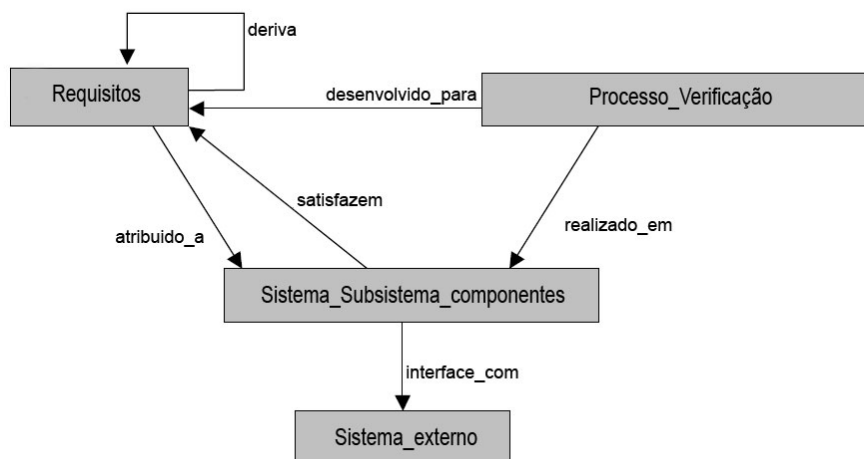


Figura 7 - Modelo de rastreabilidade *Low-End*. (Ramesh; Jarke, 2001)

High-End : Se empenham na organização da rastreabilidade e fazem o uso prolongado da mesma como parte integrante do sistema, definindo políticas de desenvolvimento. Assim eles capturam as relações entre os artefatos criados durante todo o desenvolvimento do software e o relacionamento entre os artefatos e *rationale*. Para maior apoio a sua rastreabilidade é normal que o mesmo personalize as ferramentas de acordo com suas necessidades. O modelo High-End é dividido em quatro partes: a) gerenciamento de requisitos, b) gerenciamento de *rationale*, c) processo de verificação, d) alocação no projeto, a seguir iremos mostrar as características de cada um.

- a) Com o gerenciamento os requisitos podem ser rastreados através de seu ciclo de vida provendo aos desenvolvedores uma visão se o sistema suporta os fatores críticos de sucesso. Na figura 8 temos a representação deste submodelo.
- b) O gerenciamento de *rationale* mantém as informações sobre quais decisões são tomadas para resolver os problemas e conflitos através do ciclo de vida do sistema para assegurar que os requisitos sejam satisfeitos.
- c) Projeto mostra a relação entre os requisitos e os componentes do projeto.
- d) O processo de verificação é o responsável por certificar que todos os requisitos estão sendo implementados e se estão corretos e identificar quais as mudanças que são necessárias para atingir os objetivos.

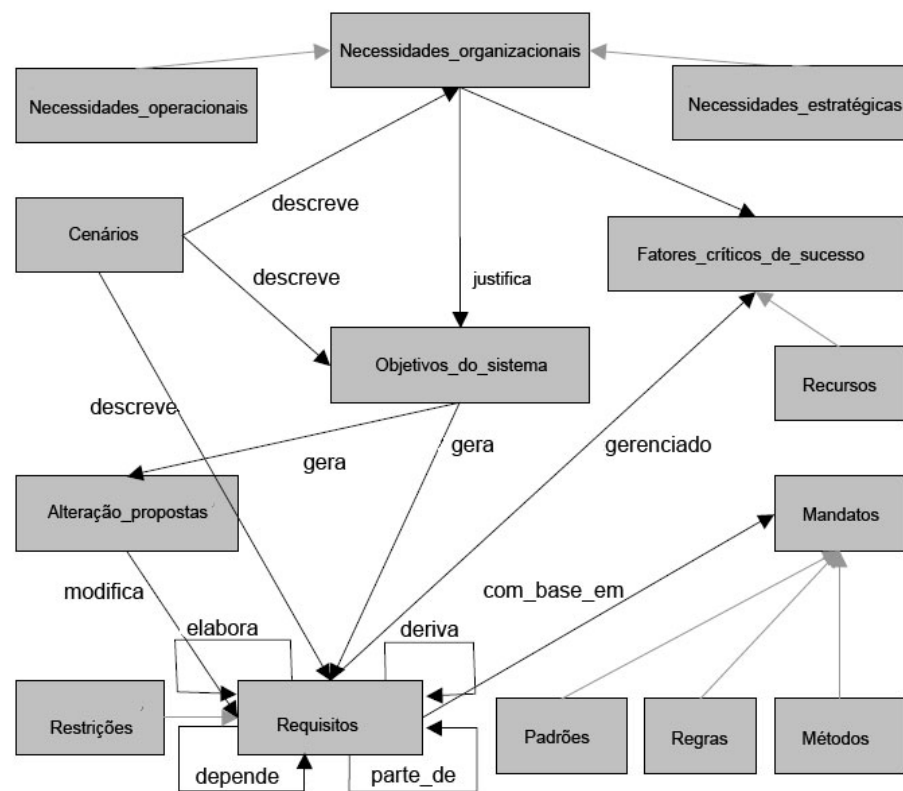


Figura 8 - Modelo de rastreabilidade High-End. (Ramesh; Jarke, 2001)

Como indicado por (Arkley *et al*, 2006) principal fator que proibi a ampla e efetiva utilização da rastreabilidade em escala industrial e a capacidade de manter as relações de rastreabilidade que poderiam apoiar as formas de análise de custo e eficácia de algumas maneiras. A diversidade de artefatos que são gerados durante o

desenvolvimento do software e a falta de interoperabilidade entre as ferramentas que são usadas para construir e gerenciar os artefatos torna a captura das relações da rastreabilidade custosas e cria a percepção que os benefícios da rastreabilidade não são justificáveis.

3.1 MÉTODOS DE RASTREABILIDADE

Atualmente existem vários métodos de rastreabilidade de software como: automático, semi-automático e manual, o qual iremos usar no nosso projeto. A seguir iremos explicar um pouco desses métodos. (Spanoudakis, 2005)

O método automático baseia-se na recuperação de informações, regras de rastreabilidade. Mas esse método ainda não está maduro para todos os tipos de links de rastreabilidade, o que faz com que crie links falsos, degradando assim o desempenho. Uma possível maneira de gerar a rastreabilidade automática seria padronizar o vocabulário que pode ser usado para especificar o modelo do sistema no domínio da aplicação. Outra possível seria a geração baseado na ontologia, onde poderia provê especificações formais de aspectos comuns do software e seu domínio. (Spanoudakis, 2005)

O método semi-automático requer que o usuário defina alguns links para que possam ser tomados como base para os próximos, e ocorre o mesmo que acontece com método automático, ou seja, ainda não está maduro para ser adotado em larga escala. O método manual de rastreabilidade traz mais informações para o processo de rastreabilidade, mas também é mais suscetível a erro, requer mais tempo e é completo. (Spanoudakis, 2005)

3.2 RASTREABILIDADE DE REQUISITOS E ARTEFATOS

Segundo Gotel (1994), a rastreabilidade de requisitos é a habilidade de descrever e seguir o ciclo de vida do requisito, em ambas as direções, para frente e para trás, desde sua origem, através de seu desenvolvimento e especificação, para sua implementação e uso, através de períodos de contínuo aperfeiçoamento e iteração em qualquer dessas fases.

A rastreabilidade de requisitos é uma forma de entender e gerenciar as informações dos requisitos visando compreender a origem dos requisitos, gerenciar e avaliar o impacto das mudanças, verificar se todos os requisitos estão satisfeitos pela implementação e verificar se realmente o que está especificado está sendo feito. (Stehle, 1990)

A rastreabilidade não pode ser somente aplicada aos requisitos, podem ser aplicadas também aos artefatos de software. No decorrer do desenvolvimento de um software são produzidos diversos artefatos, como: documentos de requisitos de projeto; código fonte; diagramas; logs de erros; etc. CMMI (2001)

A quantidade de artefatos gerados durante o ciclo de vida do software tende a aumentar em sistemas maiores e mais complexos. Um artefato pode ser definido como sendo qualquer produto tangível resultado das várias etapas de desenvolvimento de um software. Um artefato pode ser, por exemplo: um requisito, um caso de uso.

Através da rastreabilidade de artefatos é possível conhecer todo o ciclo de vida de um artefato no processo de engenharia de software em ambas as direções. (Finkelstein, 1994) Essa informação se torna necessária quando forem feitas mudanças nos requisitos, manutenções no código, reutilização e também medir o impacto que o sistema sofrerá com a mudança. (Antoniol, 2002)

Os links de rastreabilidade entre os artefatos podem ser mantidos por qualquer pessoa envolvida no desenvolvimento do software, é mais indicado que os próprios desenvolvedores façam esses links, mas infelizmente as ferramentas existentes no mercado ainda não suprem essa necessidade e como esse trabalho é extenso, sem a ajuda de uma ferramenta, provavelmente não poderá ser feito e influenciando na baixa qualidade dos sistemas produzidos. (Tortora *et al*, 2007)

Algumas ferramentas no mercado como: DOORS, RequisitePro e CaliberRM, possibilitam ao desenvolvedor definir relações ou links entre requisitos individuais e entre requisitos e outros elementos do sistema, definir diferentes tipos de atributos para diferentes tipos de requisitos e definir atributos para requisitos individualmente, integração com outras ferramentas de teste e gerência de projeto. (Wiegers, 2003)

4. PADRÃO XML

XML é uma variação restrita da SGML(standard generalized markup language), que descreve uma classe de objetos de dados que tem estrutura aninhada e é usada para marcar os dados contidos nos objetos (Oracle XML), também utilizada para permitir troca de informações de forma estruturada, permitindo que os desenvolvedores transportem dados na internet integrando diferentes aplicações(Babylon).

Existem vários esquemas (regras de validação) que podem ser utilizados nos arquivos XML, nos usaremos o esquema “*xmlbasedsrs*”. Este esquema foi criado para facilitar a documentação dos requisitos. Com este esquema pode-se representar os requisitos não-funcionais e funcionais, utilizando a notação XML. Este padrão é composto de algumas tags, veja Tabela 3.

Tabela 3 – Tags XMLBASEDSRS (Arquivo DTD do XMLBASEDSRS)

TAG	DESCRIÇÃO
Project	Descrição do projeto.
Title	Titulo do projeto ou caso de uso.
Version	Versão do projeto.
DI	Adiciona uma introdução ou outra informação complementar.
usecasetree	Contem informações sobre os atores e suas interações com os casos de uso.
Actor	Defini os atores que vao interagir com o software.
Usecase	Caso de uso
Nfrs	Requisitos não-funcionais
Description	Adiciona uma descrição
Requirement	Requisitos do sistema
Reqref	Requisito referenciado
Action	Ação do caso de uso
Reaction	Reação do caso de uso
Status	Defini o status do requisito
Id	Defini um ID para um caso de uso ou requisito.

Na Figura 9 temos um exemplo de um arquivo XML utilizando o *schema* XMLBASEDSRS, nas linhas 5 a 12 são acrescentadas informações sobre o projeto, nas linhas 13 a 22 as informações a respeito dos requisitos não-funcionais e nas linhas 23 a 31 as informações referentes aos requisitos funcionais, que são representados como casos de uso.

```

1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <!DOCTYPE srs SYSTEM "srs.dtd">
3
4 <srs>
5   <project>
6     <title>SOFTWARE MONOGRAFIA 2009.1</title>
7     <version>1.0.0</version>
8     <dl>
9       <dt>SOFTWARE MONOGRAFIA 2009.1</dt>
10      <dd>O PROGRAMADOR DEVE CONSEGUIR ASSOCIAR E DESASSOCIAR AS CLASSES AOS REQUISITOS</dd>
11    </dl>
12  </project>
13  <nfrs>
14    <dl>
15      <dt>USABILIDADE</dt>
16      <dd>
17        <requirement id="R01" status="Val">
18          <description>TODAS AS PÁGINAS DEVEM TER O FUNDO BRANCO</description>
19        </requirement>
20      </dd>
21    </dl>
22  </nfrs>
23  <useCaseTree>
24    <actor id="A01" name="PROGRAMADOR" ></actor>
25    <useCase id="C01" actors="A01">
26      <title>ASSOCIAR CLASSE</title>
27      <requirement id="R04" status="New">
28        <description>ASSOCIAR CLASSE AO REQUISITO</description>
29      </requirement>
30    </useCase>
31  </useCaseTree>
32 </srs>

```

Figura 9 – Exemplo de arquivo XML utilizando *schema XMLBASEDSRS*

5. PROJETO

Como é constante a mudança dos requisitos do software muitas vezes os desenvolvedores enfrentam problemas por não saberem quais artefatos (código fonte) estão associados aos requisitos, tendo assim maior chance de ocorrer erros e prejudicar a qualidade final do produto.

Com base no resultado do trabalho desenvolvido nos criamos um Add-In para Visual Studio 2008 chamado **TraceFact-In**. Este Add-In foi desenvolvido para auxiliar as pessoas envolvidas no processo de criação de software a ter uma forma de associar os artefatos (código fonte) com os requisitos descritos no documento de especificação, que é escrito utilizando a ferramenta RequisitePro. Para que os requisitos especificados possam ser importados para a ferramenta o projeto no RequisitePro deve ser criado utilizando o banco de dados SQL Server.

O objetivo desta ferramenta é possibilitar que os desenvolvedores criem links de rastreabilidade entre os requisitos e os artefatos gerados durante o processo de desenvolvimento de software armazenando o mesmo em um arquivo XML. A ferramenta deve dar aos programadores a opção de associar e desassociar os artefatos aos requisitos, gerar relatórios e a permitir a importação dos requisitos especificados no RequisitePro para um arquivo XML utilizando *schema XMLBASEDSRS*.

A motivação de fazer esta ferramenta foi a falta de ferramentas no mercado que possibilitem a rastreabilidade entre os requisitos e o código fonte, pois as existentes possibilitam ao desenvolvedor utilizar a rastreabilidade de requisitos e de alguns tipos artefatos, mas não do código fonte.

5.1 AMBIENTE

Esta ferramenta foi desenvolvida e testada no hardware e software:

- Processador: Turion(tm) X2 Mobile Technology TL-50 1.6 GHz
- Memória: 1,5 Gb DDR2
- Windows 7 Ultimate (Build 7100), 32-bit, Inglês
- Visual Studio 2008 Team System versão 9.0.21022.8 RTM

5.2 REQUISITOS

REQUISITOS FUNCIONAIS	
CÓDIGO	DESCRIÇÃO
RF1	Permitir a associação do artefato com o requisito
RF2	Permitir a desassociação do artefato com o requisito
RF3	Permitir a geração de relatórios das associações
RF4	Permitir a consulta da associação dos artefatos
RF5	Permitir a consulta da associação dos requisitos
RF6	Permitir a importação dos requisitos levantados no RequisitePro, através do banco de dados Sql Server

REQUISITOS NÃO FUNCIONAIS		
CÓDIGO	DESCRIÇÃO	
RNF1	Desempenho	Tempo médio de resposta inferior a dois (2) segundos por operação.
RNF2	Usabilidade	Deverá ser simples e fácil de ser executado.
RNF3	Operação	Computador com no mínimo 512GB de memória RAM, Visual Studio 2008 instalado e Framework .NET 3.5

5.2 ANÁLISE DA FERRAMENTA

O propósito desta ferramenta é complementar o que pode ser feito no IBM Requisite Pro, no mesmo podemos ter um gerenciamento total dos requisitos do sistema e dos casos de uso, fornece análise de impacto aos envolvidos. O que nosso software propõe é importar os requisitos especificados utilizando o Requisite Pro ao Visual Studio 2008 através de um Add-In e possibilitar ao desenvolvedor manter a rastreabilidade dos artefatos (métodos) com os requisitos, que poderá ser visualizada através de relatórios e tela de consulta que os próprios poderão gerar a qualquer momento.

5.2.1 DIAGRAMA DE CASO DE USO

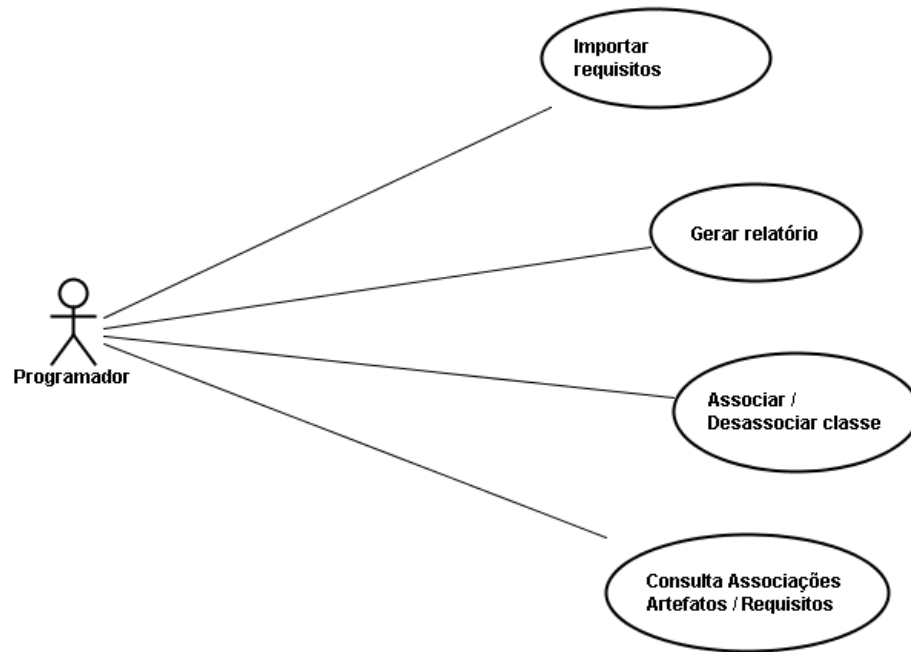


Figura 10 – Diagrama de caso de uso

Atores: Programador

5.2.2 ESPECIFICAÇÃO CASOS DE USO

Numero: UC-01

Caso de uso: Exibir opções menu

Descrição: Este caso de uso exibi as opções do menu para o usuário.

Atores: Programador

Pré-condição: Ter uma solução do Visual Studio 2008 aberta.

Fluxo Principal:

1. Abrir o Visual Studio 2008
2. Carregar uma solução do Visual Studio 2008
3. Abrir a ferramenta
4. O usuário escolhe a opção:
 - a. Associar / Desassociar : Ver caso de uso UC-02.
 - b. Importação: Ver caso de uso UC-03.

- c. Consultar Artefatos: Ver caso de uso UC-04
- d. Gerar Relatórios: Ver caso de uso UC-05

Numero: UC-02

Caso de uso: Associar artefato

Descrição: Este caso de uso permite que o programador associe o artefato com o requisito.

Atores: Programador

Pré-condição: Ter uma solução do Visual Studio 2008 aberta.

Fluxo Principal:

1. Escolher o requisito.
2. Escolher a artefato.
3. Confirmar a associação.

Fluxo Alternativo:

1. UC-01

Numero: UC-03

Caso de uso: Desassociar artefato

Descrição: Este caso de uso permite que o programador exclua a associação do artefato com o requisito.

Atores: Programador

Pré-condição: Ter uma solução do Visual Studio 2008 aberta.

Fluxo Principal:

1. Escolher o requisito.
2. Escolher o artefato.
3. Confirmar a desassociação.

Fluxo Alternativo:

1. UC-01

Fluxo Exceção:

2. Caso não seja selecionado nenhum requisito e artefato exibirá uma mensagem de erro.

Numero: UC-04

Caso de uso: Gerar relatório

Descrição: Este caso de uso permite que o programador gere um relatório das associações.

Atores: Programador

Pré-condição: Ter uma solução do Visual Studio 2008 aberta.

Fluxo Principal:

3. Escolher o tipo de relatório
4. Gerar o relatório

Fluxo Alternativo:

1. UC-01

Fluxo Exceção:

1. Caso não exista informação exibirá uma mensagem de erro.

Numero: UC-05

Caso de uso: Consultar associação artefatos / requisitos

Descrição: Este caso de uso permite que o programador consulte as associações dos artefatos.

Atores: Programador

Pré-condição: Ter uma solução do Visual Studio 2008 aberta.

Fluxo Principal:

1. Escolher a classe
2. Escolher o método

Fluxo Alternativo:

1. UC-01

Fluxo Exceção:

1. Caso não exista informação exibirá uma mensagem de erro.

5.2.3 DER

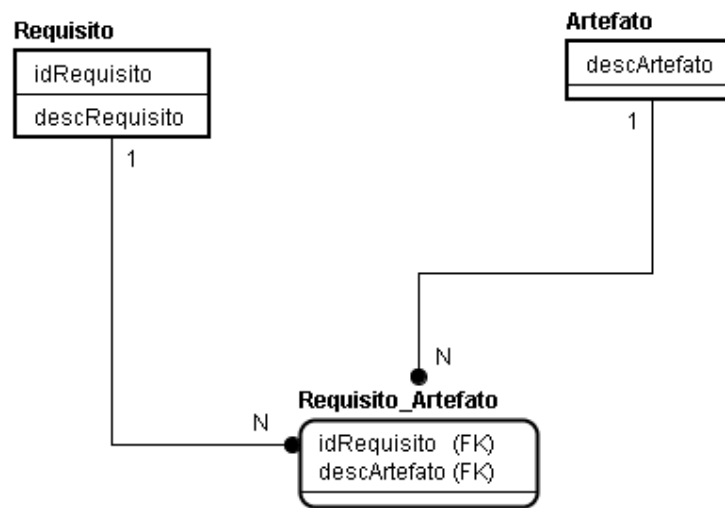


Figura 11 – DER

5.3 PROJETO TRACEFACTIN

5.3.1 ARQUITETURA

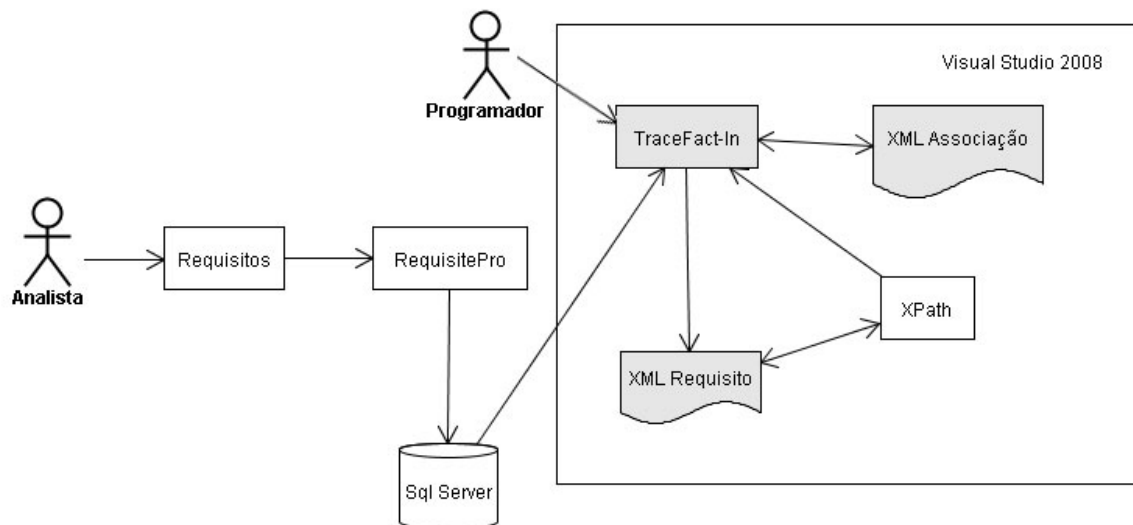


Figura 12 – Arquitetura da ferramenta

5.3.2 DIAGRAMA DE CLASSE

Nesta seção mostraremos o diagrama de classe da ferramenta proposta.

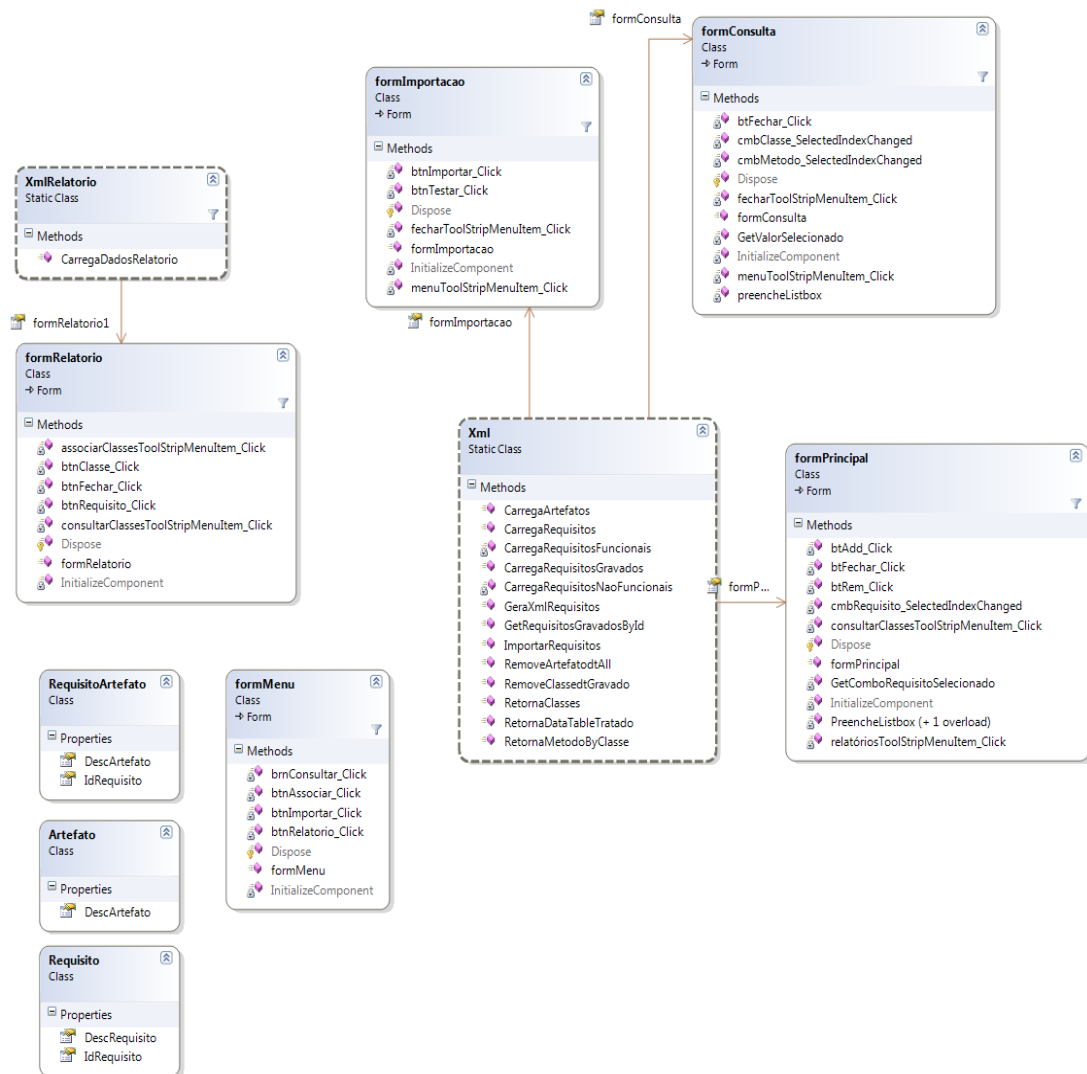


Figura 13 – Diagrama de classe

5.3.3 DIAGRAMA DE SEQUÊNCIA

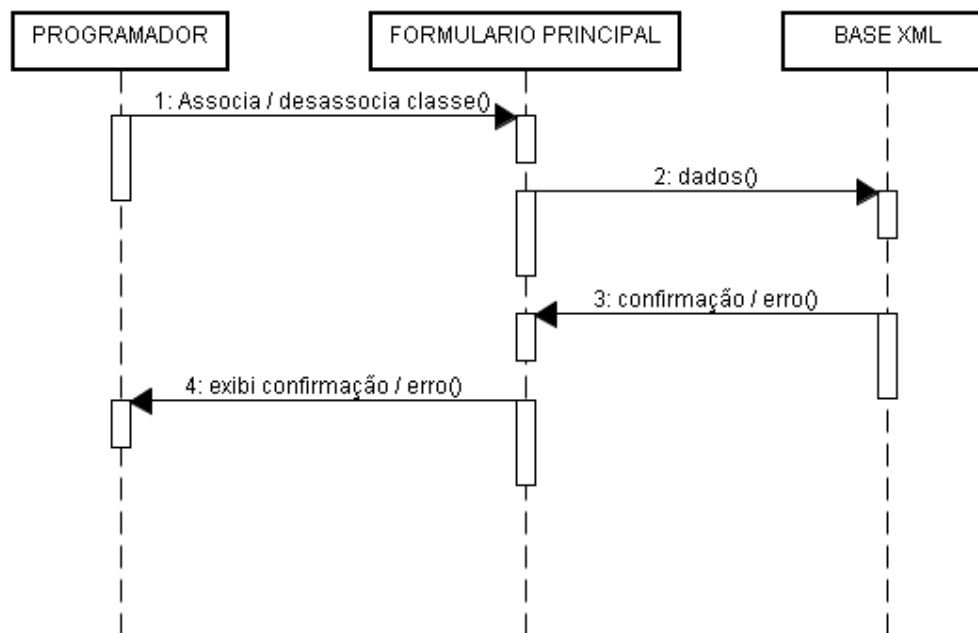


Figura 14 – Diagrama de sequencia 1

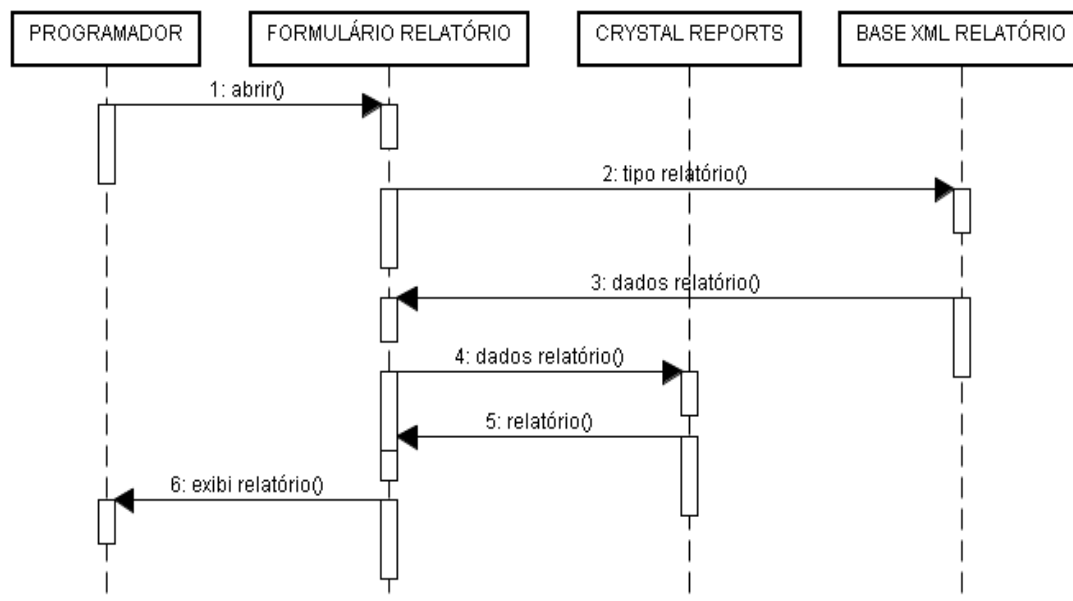


Figura 15 – Diagrama de sequencia 2

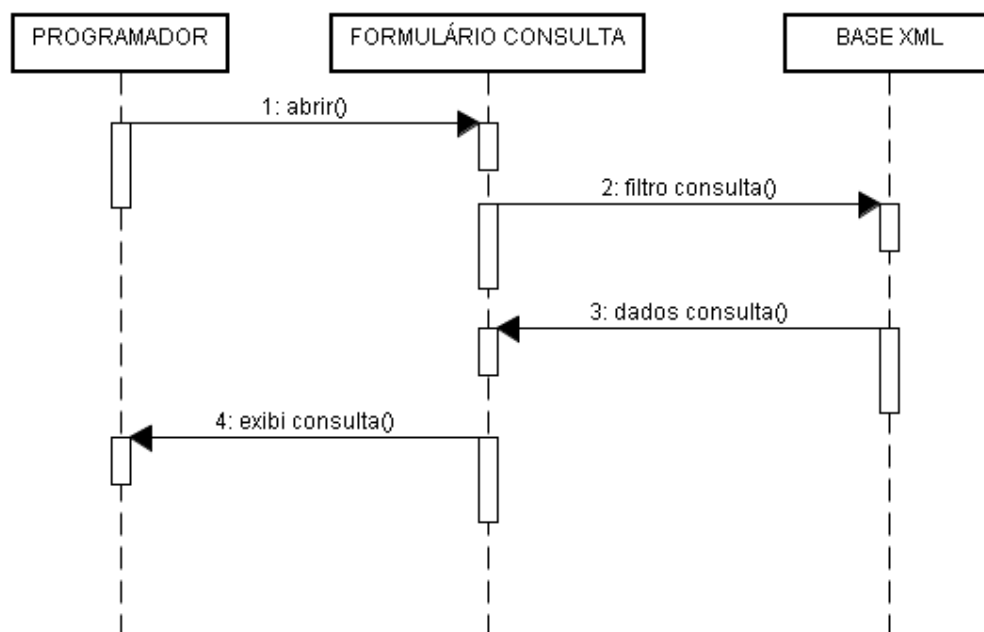


Figura 16 – Diagrama de sequencia 3

O diagrama da Figura 16 mostra o fluxo da consulta das associações dos metodos e requisitos.

5.4 USO DA FERRAMENTA

Para conseguir utilizar a ferramenta inicialmente precisa-se abrir o Visual Studio 2008, carregar uma solução, depois ir ao menu *tools* e clicar em *TraceFactIn*. Após fazer estes passos irá aparecer o formulário principal da ferramenta. Veja Figura 17.

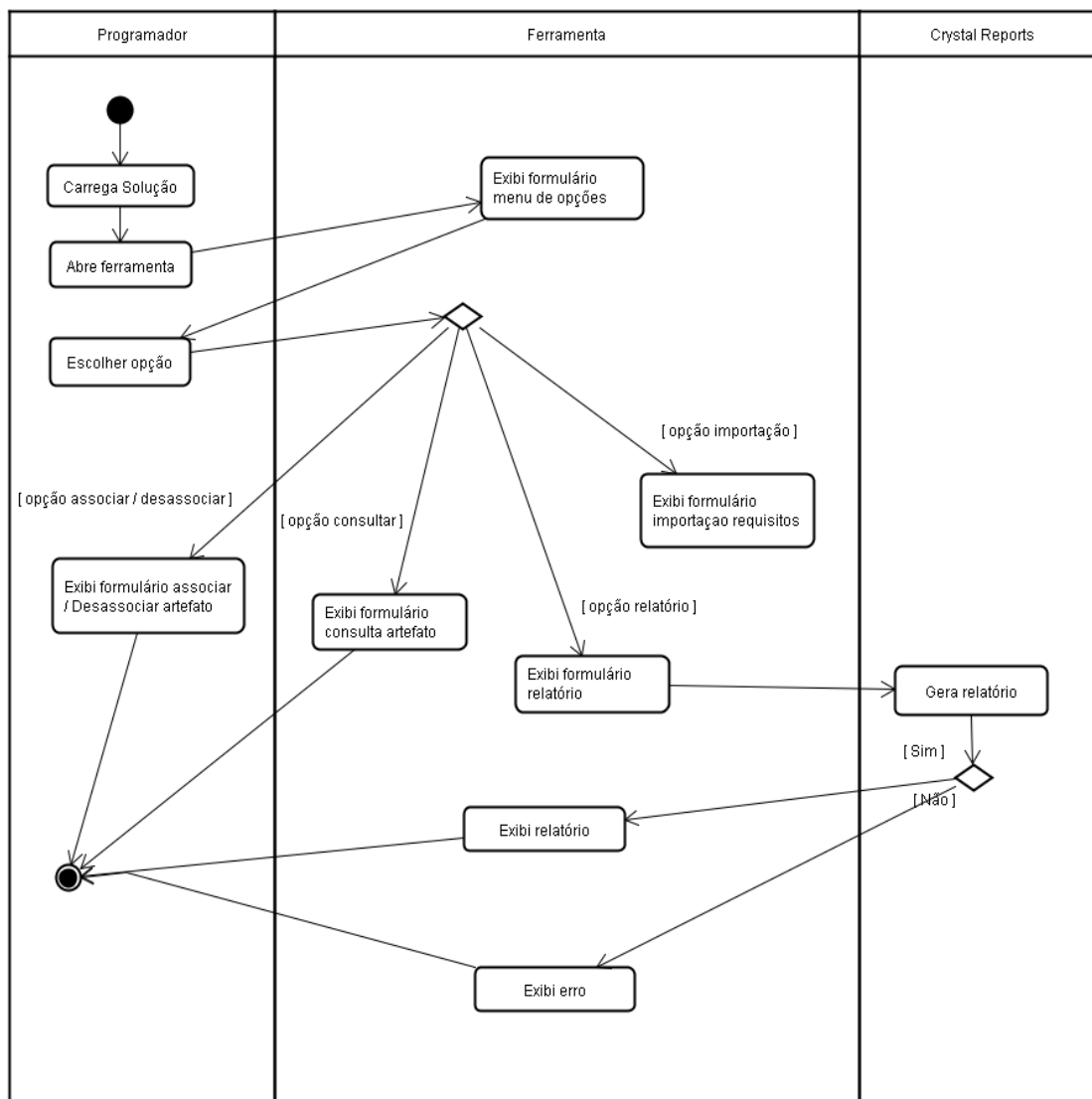


Figura 17 – Diagrama de atividade

5.4.1 MENU

A figura 18 apresenta a tela inicial da ferramenta, onde encontramos o menu de opções. Para selecionar uma funcionalidade basta clicar na opção correspondente e um novo formulário com a opção selecionada será exibido.



Figura 18 – Formulário menu

5.4.2 ASSOCIAR / DESASSOCIAR ARTEFATO

A figura 19 ilustra a tela de associar / desassociar artefatos, para conseguir associar ou desassociar artefatos deve-se primeiramente escolher o requisito, quando o requisito for selecionado as duas caixas Associado e Não Associado serão preenchidas. Após fazer a escolha do artefato basta clicar em Associar ou Desassociar e uma mensagem de confirmação será exibida. Caso não selecione nenhum requisito ou artefato e clique em Associar ou Desassociar, aparecerá uma mensagem de erro.

A imagem mostra uma janela de software intitulada "ProjetoFinal". No topo, há uma barra de título com o nome da janela e botões de controle padrão. Abaixo, a seção "Ações" contém um menu suspenso rotulado "Requisito", no qual a opção "R04 - ASSOCIAR CLASSE AO REQUISITO" está selecionada. A interface é dividida em duas áreas principais: "Não Associado" à esquerda e "Associado" à direita. A caixa "Não Associado" contém uma lista com os itens "PROJETO.CLASSETESTE.Metodo1" e "PROJETO.CLASSETESTE.Metodo2". A caixa "Associado" está atualmente vazia. Entre essas duas áreas, há dois botões: "Associar" e "Desassociar". No canto inferior direito da janela, há um botão "Fechar".

Figura 19 – Formulário de associar / desassociar artefato

5.4.3 GERAR RELATÓRIO

Para acessar esta funcionalidade da ferramenta deve-se acessar o menu Ações e clicar em Relatórios, após o formulário aberto irão aparecer duas opções de relatório. Para gerar o relatório deve-se clicar em um dos dois tipos. Caso o relatório não possa ser gerado, será exibida uma mensagem de erro, caso contrário o relatório será exibido. Veja Figura 20.

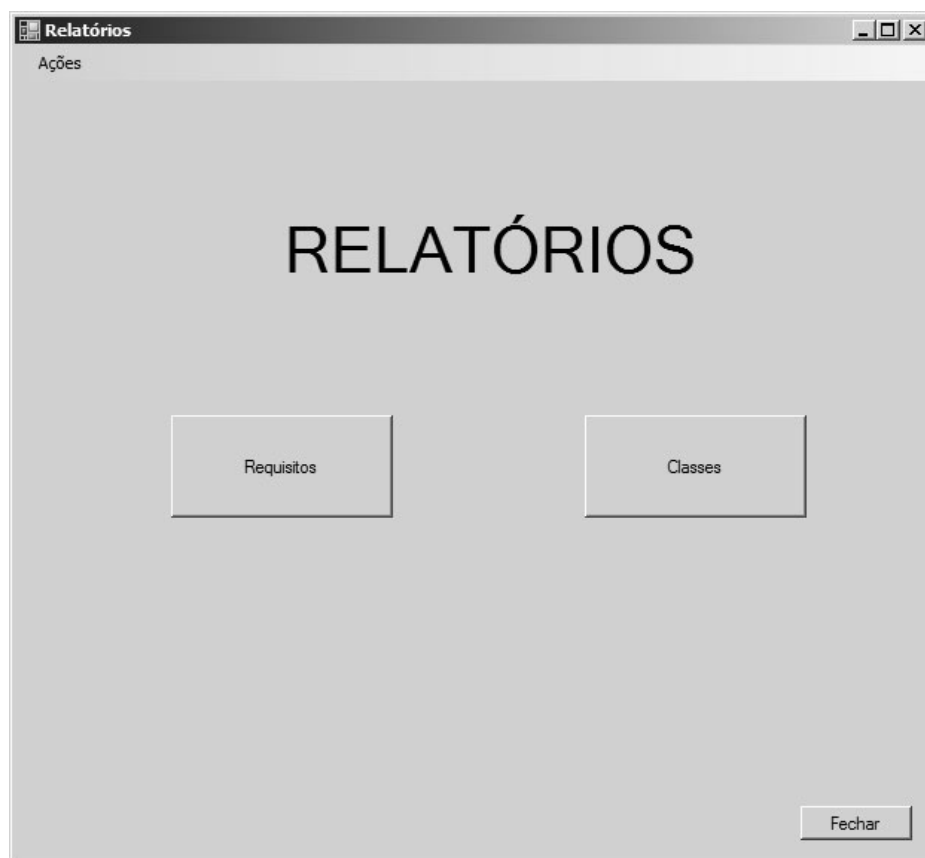
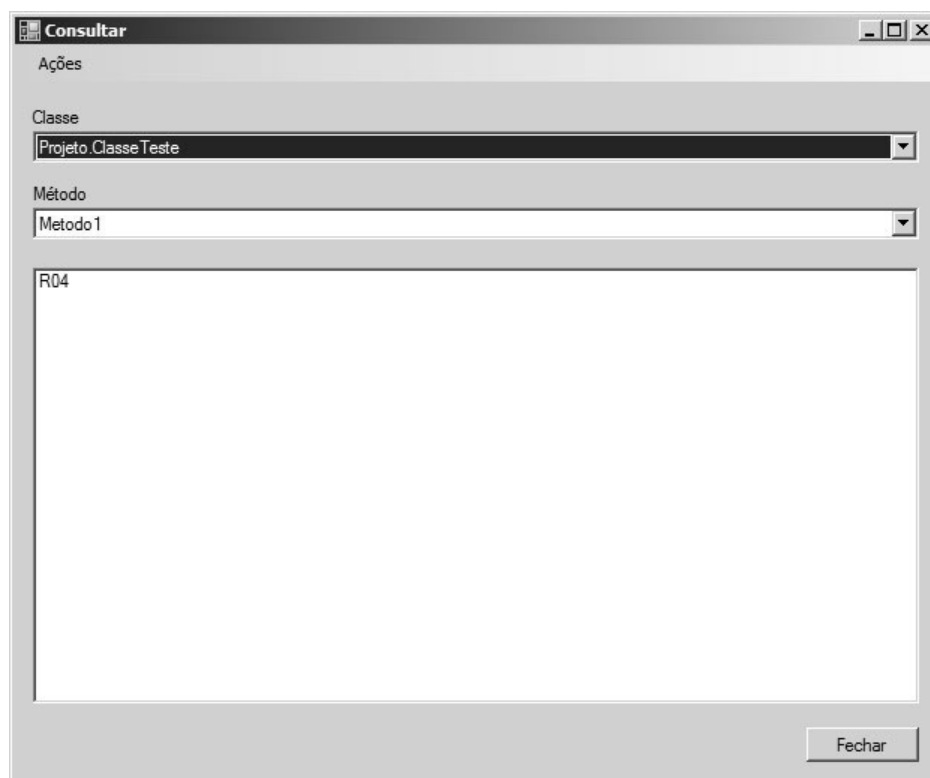


Figura 20 – Formulário Gerar Relatório

5.4.4 CONSULTAR ARTEFATOS

Para acessar esta funcionalidade da ferramenta deve-se acessar o menu Ações e clicar em Consultar Artefatos. Para efetuar a consulta deve-se escolher uma classe, quando a classe for escolhida e a mesma tenha métodos, os mesmos apareceram na lista Método, depois se escolhe o método e caso o mesmo esteja associado a algum requisito, a informação aparecerá na caixa Associações. Veja Figura 21.

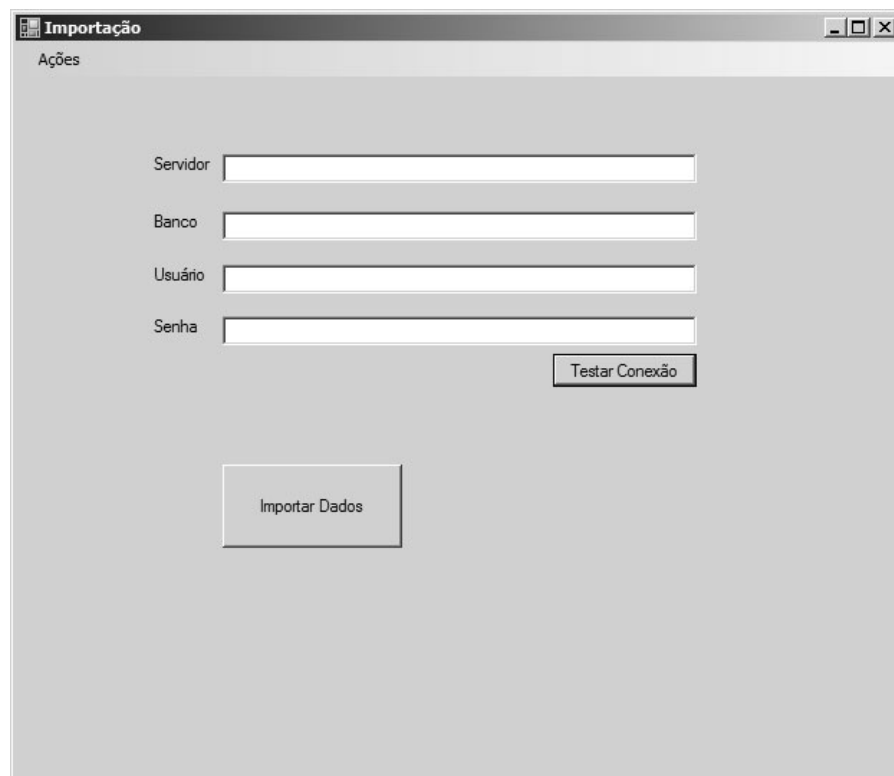


The image shows a software window titled "Consultar". It contains a section labeled "Ações" at the top. Below it, there is a "Classe" label followed by a dropdown menu showing "Projeto.ClasseTeste". Underneath that is a "Método" label followed by a dropdown menu showing "Metodo1". Below the dropdowns is a large text area containing the text "R04". At the bottom right of the window is a button labeled "Fechar".

Figura 21 – Formulário Consultar Artefatos

5.4.5 IMPORTAÇÃO REQUISITOS

Para acessar esta funcionalidade da ferramenta deve-se acessar o menu Ações e clicar em Importar. No formulário é necessário informar os dados para acessar o banco de dados do RequisitePro, no campo Servidor põe-se o endereço do servidor de banco de dados, no campo Banco põe-se o nome do banco de dados, no campo Usuário põe-se o usuário para autenticar no banco de dados e no campo senha põe-se a senha para autenticar no banco de dados. Após inserir os dados deve-se clicar em Testar Conexão para testar se a comunicação com o banco de dados esta sendo estabelecida, caso obtenha sucesso uma mensagem de sucesso será exibida, caso contrário será exibida uma mensagem de erro. Para importar os requisitos do RequisitePro deve-se clicar em Importar Dados, caso a importação seja realizada com sucesso uma mensagem de sucesso será exibida e caso contrário uma mensagem de erro será exibida.



The image shows a software window titled "Importação". Inside the window, there is a section labeled "Ações". Below this section, there are four text input fields labeled "Servidor", "Banco", "Usuário", and "Senha" arranged vertically. To the right of the "Senha" field, there is a button labeled "Testar Conexão". Below these fields, there is a larger button labeled "Importar Dados".

Figura 22 – Formulário de implantação

5.5 AVALIAÇÃO DA FERRAMENTA

A ferramenta foi testada utilizando um projeto de pequeno porte, que faz o controle de clientes e envio de mala direta. O projeto tem 11 requisitos funcionais e 3 requisitos não funcionais, é composto por uma solução do Visual Studio 2008 com 2 projetos associados, sendo um *Website* e outro *ClassLibrary*.

A ferramenta mostrou-se estável durante todo o teste, conseguindo contemplar todos os requisitos funcionais e não-funcionais descritos anteriormente. Nos testes foi possível verificar que a ferramenta consegue apresentar informações importantes aos desenvolvedores de forma simples e clara através de telas de consulta e relatórios, as informações fornecidas pela rastreabilidade de artefatos foram satisfatórias, pois foi possível analisar melhor o impacto que as alterações nos requisitos iriam ter no projeto e ampliar a visão sobre a reusabilidade de código.

Existem alguns pontos que podem ser melhorados na ferramenta como a criação de mais opções para o usuário associar e consultar os artefatos e requisitos, suporte a outros Servidores de Banco de Dados e melhorar a apresentação dos dados e dos formulários.

Como a ferramenta foi testada em um projeto pequeno, com requisitos simples há o risco que a mesma não funcione satisfatoriamente com projetos que tenham requisitos mais complexos.

6. CONCLUSÃO

Este trabalho teve como objetivo demonstrar a rastreabilidade de artefatos e desenvolver uma ferramenta que possibilite a rastreabilidade dos requisitos com os métodos das classes geradas. Através de conceitos de engenharia de software e engenharia de requisitos, mostramos as suas vantagens e como servem de suporte para a rastreabilidade de artefatos.

Os resultados obtidos a partir da ferramenta permitiram concluir que a rastreabilidade de artefatos é útil no processo de desenvolvimento de software e objetivo de integrar a ferramenta com o Requisite Pro para trazer informações relevantes no processo de desenvolvimento de software aos desenvolvedores foi alcançado, aumentando a segurança ao se fazer alterações nos requisitos.

Com a ferramenta proposta expandimos o controle dos desenvolvedores aos métodos, tornando a análise de impacto mais concreta, pois sem esta ferramenta não seria possível saber quais os métodos iram ser afetados pela mudanças sugeridas sendo evolução ou correção. A principal dificuldade desse trabalho foi a de encontrar ferramentas permitissem ao desenvolvedor associar os requisitos aos métodos, para que pudessem servir de base pra o desenvolvimento da solução proposta.

Como proposta de melhoria desta ferramenta sugerimos o estudo de uma melhor de integração entres o Requisite Pro e a ferramenta proposta, a criação de padrões para a identificação automática dos links de rastreabilidade e a extensão da recuperação dos métodos em outros tipos de projeto que não sejam *ClassLibrary*.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTONIOL, G., et al. **Recovering traceability links between code and documentation.** *Transactions on Software Engineering*. Oct de 2002, Vol. 28, pp. 970-983.

BAUER, Friedrich Ludwig, **NATO Software Engineering Conference**. Alemanha, 1968. Disponível em: <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>, 10 mar 2009.

BABYLON, **Dicionário da língua portuguesa**. Disponível em: <http://www.babylon.com.br>, 13 maio 2009.

CLOUSE, Dennis M. Ahern; TURNER, Richard Aaron. **CMMI Distilled: A Practical Introduction to Integrated Process Improvement**. Boston : Addison-Wesley Longman Publishing Co., Inc., 2001.

DORFMAN, M; TAYER, R.H. **Standards, Guidelines, and Examples on System and Software Requirements Engineering**. IEEE Computer Society Press, 1990.

EDWARDS, M.; HOWELL, S. **A methodology for system requirements specification and traceability for large realtime complex systems**. Technical report, Naval Surface Warfare Center, 1991.

FIKSEL, J.D. **New requirements management software supports concurrent engineering**. CimFlex Teknowledge Corporation, Washington, DC, 1994.

FINKELSTEIN, Orlena Gotel; Anthony. **An analysis of the requirements traceability problem.** *International Conference on Requirements Engineering*. 1994, pp. 94-101.

GREENSPAN, S.; MCGOWAN, C. **Structuring software development for reliability.** *Microelectronics and Reliability* 17.

HAMILTON V.L.; BEEBY, M.L. **Issues of traceability in integrating tools**. Proc. Colloquium IEE Professional Group C1, London 1991.

IEEE. **IEEE Standard Glossary of Software Engineering.** *IEEE Std 610.12- 1990*. 10 Dec 1990.

KOSCIANSKI, André; *et tal.* **Qualidade de Software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. São Paulo: Novatec Editora, 2006.

MARCUS A.; MALETIC J.I. **Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing**, ICSE, 2003

MCGLAUGHLIN, R. **Some Notes on Program Design**, *Software Engineering Notes*, vol. 16, no. 4, October 1991, pp. 53–54.

Oracle XML, **O Manual Oficial**, 2ª edição, 2001

PAULA FILHO, Wilson de Pádua. **Engenharia de Software: Fundamentos, Métodos e Padrões**. 2. ed. Rio de Janeiro: LTC, 2003.

PINHEIRO, F.; GOGUEN, J. **An object-oriented tool for tracing requirements**. IEEE Software, March 1996.

PRESSMAN, Roger S., **Engenharia de Software**. São Paulo, Pearson Education do Brasil, 1995.

PRESSMAN, Roger S., **Engenharia de Software**. São Paulo, Pearson Education do Brasil, 2005.

RAMESH, B.; EDWARDS, M. **Issues in the development of a model of requirements traceability**. Proc. International Symposium on Requirements, San Diego, CA, January 1993.

RAMESH, B.; JARKE, M. **Toward Reference Models for Requirements Traceability**
IEEE Transactions on Software Engineering, Vol. 27, No. 1, January 2001

RICHARDSON, Julian; GREEN, Jeff. **Automating Traceability for Generated Software Artifacts**, pp.24-33, 19th IEEE International Conference on Automated Software Engineering (ASE'04), 2004

SOMMERVILLE, Ian. **Engenharia de Software**. 6ª edição, 2003

STEHLE, G. **Requirements traceability for real-time systems**. Proc. EuroCASE II, London 1990.

SWEBOK, **Guide to the software Engineering Body of Knowledge**. Disponível na internet : <http://www2.computer.org/portal/web/swbok/htmlformat>, 10 mar 2009.

TORTORA, Genoveffa; DE LUCIA, Andrea; FASANO, Fausto; OLIVETO, Rocco. **Recovering traceability links in software artifact management systems using information retrieval methods**. *ACM Trans. Softw. Eng. Methodol.* 2007, Vol. 16.