Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Trace Tracker: um sistema para gerenciamento de rastreabilidade

Bruno Farache

TRABALHO DE GRADUAÇÃO

Recife, março de 2007

Universidade Federal de Pernambuco
Centro de Informática

Bruno Farache

Trace Tracker: um sistema para gerenciamento de rastreabilidade

Monografia apresentada ao Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Alexandre Marcos Lins de Vasconcelos

Recife, março de 2007.

Agradecimentos

À minha família, as pessoas mais importantes na minha vida. Ao meus amigos e Camilla, pelo companheirismo e compreensão.

Ao professor Alexandre Vasconcelos e Sandro Oliveira por terem me ensinado muito durante este trabalho.

Resumo

Diversos artefatos são produzidos durante o desenvolvimento de um software e existe a necessidade de se ter uma visão sobre as dependências existentes entre esses artefatos.

Várias ferramentas de apoio ao desenvolvimento de software produzem artefatos de mesma natureza contudo com formatos diferentes, o que possibilita apenas um controle de rastreabilidade manual.

Este trabalho se propõe a estudar maneiras de inferir, quando possível, as dependências automaticamente e semi-automaticamente e apresentará o sistema Track Tracer que adotará as melhores técnicas e padrões estudados.

SUMÁRIO

Lista de figuras

Lista de tabelas

1. Introdução

Durante o processo de desenvolvimento de softwares vários artefatos são produzidos, tais como: requisitos, casos de testes, algoritmos e casos de uso. Uma informação bastante relevante para os envolvidos na produção de um software é a correlação existente entre esses vários artefatos, ou seja, se faz necessário manter continuamente uma rastreabilidade atualizada entre os vários artefatos produzidos. Tal rastreabilidade traz vários benefícios para a qualidade e eficiência no desenvolvimento de softwares.

1.1. Motivação

Existem algumas barreiras que não encorajam o controle da rastreabilidade durante a produção de um software, uma delas é o fato de que os artefatos muitas vezes não são estruturados e sim textuais, impedindo que dependências entre esses artefatos sejam automaticamente inferidas, em conseqüência disto o controle da rastreabilidade tem que ser feito manualmente, tornando-se um trabalho árduo e aumentando a probabilidade de existirem inconsistências. Além disso, as pessoas responsáveis por manter a rastreabilidade muitas vezes não são as mesmas que são beneficiadas por essa manutenção, dificultando a percepção da relevância desta prática.

Outra barreira é que os artefatos são produzidos em ferramentas de apoio bastante distintas, fabricadas por diversas empresas. Muitas vezes as ferramentas não produzem artefatos que seguem um padrão aceito mundialmente que facilitasse a integração entre elas. Por exemplo: um caso de uso é criado em uma ferramenta, porém o caso de teste que o verifica é criado em uma ferramenta distinta, apesar de esses dois artefatos estarem intimamente relacionados, o link entre o dois não é automaticamente inferido pelas ferramentas.

Além disso, ainda não existe um modelo padrão para a representação dessas dependências, impossibilitando a integração entre gerenciadores de rastreabilidade distintos.

1.2. Objetivos

O objetivo deste trabalho é estudar o estado da arte da rastreabilidade e entender as principais causas das organizações não praticarem esta atividade amplamente.

Outro objetivo é propor um software para gerenciamento de rastreabilidade chamado Trace Tracker. Este sistema tem como objetivo suprimir as barreiras encontradas pelas organizações durante a manutenção da rastreabilidade.

1.3. Metodologia

A metodologia consiste em primeiramente estudar mais a fundo o significado da prática de rastrear artefatos nas organizações e entender seus maiores benefícios e problemas. A partir daí haverá uma análise das ferramentas existentes e compará-las, extraindo suas melhores e piores características.

Como produto do que foi aprendido será modelado e desenvolvido o Trace Tracker.

1.4. Estrutura do trabalho

- Capítulo 2 – uma visão geral do problema;
- Capítulo 3 – comparativo entre as ferramentas existentes;
- Capítulo 4 – proposta e descrição do Trace Tracker;
- Capítulo 5 – conclusão e ponderações sobre o trabalho;

2. Rastreabilidade: uma visão geral

2.1. Definição

Segundo a IEEE, o termo rastreabilidade *(traceability)* é definido da seguinte maneira [1990 IEEE Std. Glossary]:

"(1) The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match; (2) The degree to which each element in a software development product establishes its reason for existing; for example, the degree to which each element in a bubble chart references the requirement that it satisfies."

Ou seja, rastreabilidade é o ato de se manter uma representação das associações diretas e indiretas entre os vários artefatos produzidos durante o desenvolvimento de um software.

Através dessa representação podemos entender o grau de relação entre esses vários artefatos e também podemos observar qual a direção de dependência existente entre dois artefatos.

Artefato é definido como sendo qualquer produto tangível resultado das várias etapas de fabricação de um software [Artifact Wikipedia]. Um artefato pode ser, por exemplo: um requisito, um caso de uso, um modelo UML, o plano do projeto e até mesmo se referir ao código-fonte.

2.2. Benefícios

Durante e após o desenvolvimento de um software, os *stakeholders* em geral sentem a necessidade de se informar sobre as relações de causa e efeito entre os artefatos produzidos. O ato de rastrear traz respostas para essas perguntas e é utilizado desde a década de 70 pela Engenharia de Requisitos.

Cada perfil de *stakeholder* possui necessidades específicas [2006 Model Traceability], como por exemplo:

- O gerente de projeto quer saber se todos os requisitos foram satisfeitos e se cada componente satisfaz um requisito;
- O gerente de requisitos necessita entender a evolução das mudanças dos requisitos, quais os motivos das mudanças e impactos em outros requisitos relacionados;
- O desenvolvedor pode estimar melhor o tempo e custo necessário para implementação de uma mudança;
- Os arquitetos podem encontrar soluções alternativas para resoluções de problemas;

A manutenção da rastreabilidade é definida e obrigatória em vários padrões de qualidade de software, tais como: MIL-STD-498, IEEE/EIA 12207 e ISO/IEC 12207. [2006 Model Traceability], o que mostra a importância do seu papel na qualidade da produção de um software.

2.3. Problemas

Apesar de se saber que a manutenção da rastreabilidade traz vários benefícios à qualidade de um software, são encontrados diversos problemas que impedem uma ampla implantação desta atividade.

Um estudo [2006 Arkley, Mason e Riddle] levantou as principais barreiras encontradas nessa área. As principais causas encontradas tinham a ver com fatores humanos e processos, além da diversidade de ferramentas heterogêneas e times de desenvolvimento distribuídos.

Além da pesquisa, foi proposto um framework para gerenciamento de rastreabilidade focado no armazenamento das decisões de arquitetura e seus motivos. Em relação a esta atividade foram encontrados três principais problemas:

- Engenheiros sobrecarregados não estão dispostos a documentar cada decisão feita;
- Não basta documentar as decisões, o acesso às informações deve ser facilitado;
- Justificativas desatualizadas em relação ao sistema são perigosas;

A pesquisa focou em descobrir como os engenheiros procedem para manter a rastreabilidade e sua importância para eles.

Os problemas descobertos foram divididos da seguinte maneira:

Problemas com as ferramentas

Cada ferramenta possui formato próprio para representação dos artefatos, necessitando conhecimento específico de como utilizá-la. Isso faz com que apenas alguns profissionais tenham facilidade em usá-la.

Muitas vezes os engenheiros preferem criar referências a documentos texto em vez de manter os artefatos nas ferramentas.

Falta de treinamento nas ferramentas utilizadas também reduzem o número de pessoas capazes de usá-las eficientemente.

Organização dos dados

Documentar as decisões do projeto é uma tarefa árdua e muitas vezes resulta em duplicação de conteúdo.

Manter as informações atualizadas também é um problema, visto que muitas vezes os envolvidos estão acostumados com uma organização própria de seus documentos, fugindo do processo padronizado pela empresa.

A internet possibilitou o desenvolvimento distribuído de software, o que de certa forma piora as comunicação entre os envolvidos e gera uma grande quantidade de informação.

Outro fator que dificulta é a reutilização de componentes de software cada vez maior, criando relacionamentos implícitos.

Por fim, o estudo sugere o framework e práticas que solucionariam os problemas encontrados:

- Padronizar as representações dos artefatos para uma maior integração entre as ferramentas;
- Manter um repositório centralizado, facilitando o desenvolvimento distribuído do software;
- Reutilização das decisões feitas em projetos passados;
- Ferramenta pró-ativa que alerte possíveis mudanças na rastreabilidade quando os artefatos são modificados;
- Utilização de técnicas de mineração de dados para que seja possível descobrir quais artefatos semi-estruturados estão potencialmente relacionados;

2.4. Classificações, técnicas e representações

Classificações

As ferramentas para gerenciamento de rastreabilidade geralmente permitem três maneiras para se classificar uma relação entre dois artefatos: apenas a indicação da direção da relação; tipos pré-definidos de relações e tipos de relações customizados pelos usuários.

A maioria das ferramentas possibilitam apenas classificar a direção entre os artefatos (como no RequisitePro e CaliberRM), ou seja, o usuário tem como identificar qual artefato é dependente do outro e qual está gerando a dependência. Em geral também é possível criar uma relação bidirecional.

Manter apenas a direção do relacionamento pode ser bastante limitante, relacionamentos com significados facilitam o entendimento da rastreabilidade pelos usuários. Ferramentas como DOORS [DOORS], por exemplo, permitem aos usuários criarem novos tipos de relacionamentos que se alinhem com os processos utilizados em suas organizações.

A outra forma de lidar com a semântica dos relacionamentos é proposta por Ramesh e Jarke [2001 Ramesh and Jarke]. São fornecidos aos usuários tipos pré-definidos de relações que são comumente utilizados, tais como: “satisfaz”, “descreve”, “justifica”, “depende de” e “valida”. A vantagem dessa proposta em relação à customização de tipos pelos clientes é que não ocorre de dois usuários criarem tipos que têm a mesma semântica mas que utilizam nomes diferentes e o lado negativo é que o usuário só poderá usar os tipos padrões oferecidos pela ferramenta.

Técnicas

A técnica mais rudimentar para a manutenção da rastreabilidade é a manual. Como já foi citado, um dos maiores fatores que dificultam o gerenciamento da rastreabilidade nas organizações é o trabalho que se tem para mantê-la. As pessoas geralmente desistem de atualizar os relacionamentos

a cada mudança nos artefatos, além de surgirem vários relacionamentos falsos por causa da evolução desses artefatos.

Outra técnica é inferir relações entre artefatos através de cálculos estatísticos como proposto por Ying [2004 Ying et al.]. A probabilidade de haver uma dependência entre artefatos aumenta quando eles são modificados e submetidos em conjunto nos sistemas de gerenciamento de configuração.

Uma maneira de também automatizar o processo é através de técnicas de mineração de dados, analisando textos dos artefatos é possível encontrar identificadores para outros artefatos ou mesmo quando se identifica termos semelhantes utilizados em artefatos distintos deve haver uma relação entre eles.

Uma técnica muito eficaz para automatizar a rastreabilidade é descrita em [2006 Model Traceability]. No desenvolvimento de softwares baseado em modelos (*Model Driven Development*) o código é gerado automaticamente a partir de transformações de modelos e essa característica permite a automação da manutenção da rastreabilidade entre o que foi gerado e suas especificações.

Representações

A maneira mais comum utilizada pelos sistemas de gerenciamento de requisitos é de representar a rastreabilidade utilizando matrizes. Os requisitos ficam nas linhas e colunas e há indicações sobre a direção de dependência entre um requisito e outro. Outra representação que é disponibilizada pelo CaliberRM é através de grafos, cada requisito é um nó do grafo e as arestas são as relações. Ainda existem outros tipos de representações, tais como: *hyperlinks*, métodos formais e bancos de dados.

2.5. Aplicação e problemas em Engenharia de Requisitos

A rastreabilidade foi primeiramente utilizada pela Engenharia de Requisitos e ainda hoje é amplamente utilizada nesta área. Gotel e Finkelstein realizaram uma pesquisa [Gotel 1994] bastante detalhada sobre rastreabilidade durante a especificação de requisitos em projetos de software.

Rastreabilidade entre requisitos foi definida pelos pesquisadores da seguinte maneira:

"Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)."

Existe aí o conceito de direção de dependência, utilizando a rastreabilidade como mecanismo de descoberta de causas e conseqüências dos requisitos.

Durante a pesquisa, foram feitos experimentos empíricos através de entrevistas e questionários durante 1 ano com mais de 100 profissionais da área de desenvolvimento de software. Diversas questões foram feitas, tais como: o que as pessoas entendem por rastreabilidade de requisitos, quais as barreiras encontradas durante esse processo e se haviam sugestões para melhoria no controle do rastreamento de requisitos.

No total, cerca de 100 sistemas gerenciadores de requisitos utilizados pelos entrevistados foram analisados. Chegou-se a conclusão de que a maioria não dá suporte à rastreabilidade e quando há as diferenças são apenas cosméticas e no esforço manual que se tem para relacionar os requisitos. Foi constatado que muitas pessoas preferiram utilizar ferramentas de propósito geral (como editores de texto) para gerenciar os requisitos por causa da falta de integração e flexibilidade dos sistemas gerenciadores de requisitos utilizados.

As ferramentas utilizadas durante a especificação de requisitos foram divididas em 3 tipos:

- Ferramentas de propósito geral: sistemas que não foram projetados para gerenciar requisitos mas podem utilizados, como por exemplo, editores de textos.
- Ferramentas de propósito específico: sistemas que são utilizados apenas para especificação de requisitos.
- Workbenches: integram ferramentas utilizadas durante o processo de desenvolvimento de um software e relaciona os artefatos gerados durante todas as fases.

Durante as entrevistas foram encontradas algumas dificuldades que dificultam o desenvolvimento de uma ferramenta de gerenciamento de requisitos perfeita, abaixo algumas delas:

Falta de uma definição comum

Os entrevistados definiram “rastreamento de requisitos” de diversas maneiras de acordo com suas necessidades, então fica difícil uma ferramenta satisfazer a todas essas definições.

Problemas indiretamente relacionados

Os profissionais alegaram diversas causas que dificultam a manutenção das relações, tais como: falta de integração entre as várias ferramentas geradoras de artefatos, informações implícitas e projetos de duração longa. Foi concluído que existem diversos problemas relatados pelos entrevistados que na verdade são muito complexos e que não englobam apenas a rastreabilidade de requisitos.

Ao fim do estudo, após coleta dos dados das entrevistas, os pesquisadores descreveram quais os principais problemas e necessidades encontradas durante a manutenção de requisitos e da rastreabilidade entre eles.

- Relacionar os requisitos é visto como um custo extra, os recursos alocados, em geral, não são suficientes;
- Falta de definição dos papéis responsáveis por cada etapa;
- Sensação de que é muito trabalho necessário para pouco resultado;
- Esforços são individuais, nem todos se comprometem;
- Falta de consenso sobre o que o cliente realmente pediu;
- Requisitos mudam, mas a atualização de seus relacionamentos é esquecida;
- Apenas documentar os requisitos não é garantia de uma rastreabilidade correta;
- Pouco feedback de quais melhores práticas devem ser seguidas;

3. Descrições e comparativo de ferramentas de rastreabilidade existentes

3.1. TraceM

É um framework conceitual para automatização do gerenciamento de rastreabilidade criado por Susanne A. Sherba, Kenneth M. Anderson e Maha Faisal da Universidade do Colorado, EUA [2003 Sherba, Anderson & Faisal], na época em que o paper foi escrito não havia um protótipo implementado.

Nesse framework é possível relacionar artefatos de naturezas distintas utilizando técnicas de *Open Hypermedia Systems* [1996 Østerbye & Will] e *Information Integration* [2002 Anderson, Sherba & Lepthien].

Open Hypermedia Systems permitem a criação e a navegação de relacionamentos entre aplicações distintas e entre relacionamentos dentro de uma mesma aplicação. As informações sobre os relacionamentos são armazenadas separadamente dos artefatos e são suportados relacionamentos de tipos complexos, como por exemplo, relacionamentos com múltiplas âncoras e relações entre relacionamentos.

Information Integration provê técnicas para descoberta automática de relações entre artefatos heterogêneos produzidos por ferramentas distintas (como IDEs, processadores de textos e programas de e-mail). Ela possibilita a criação de *Translators*, que são responsáveis por importar artefatos de ferramentas externas para o TraceM, isto torna o framework bem extensível, sendo possível importar artefatos dos mais diversos formatos. Também é possível construir *Integrators* que são responsáveis pela descoberta automática de novas relações entre artefatos.

A figura abaixo mostra o fluxo de atividades ao se utilizar o TraceM:

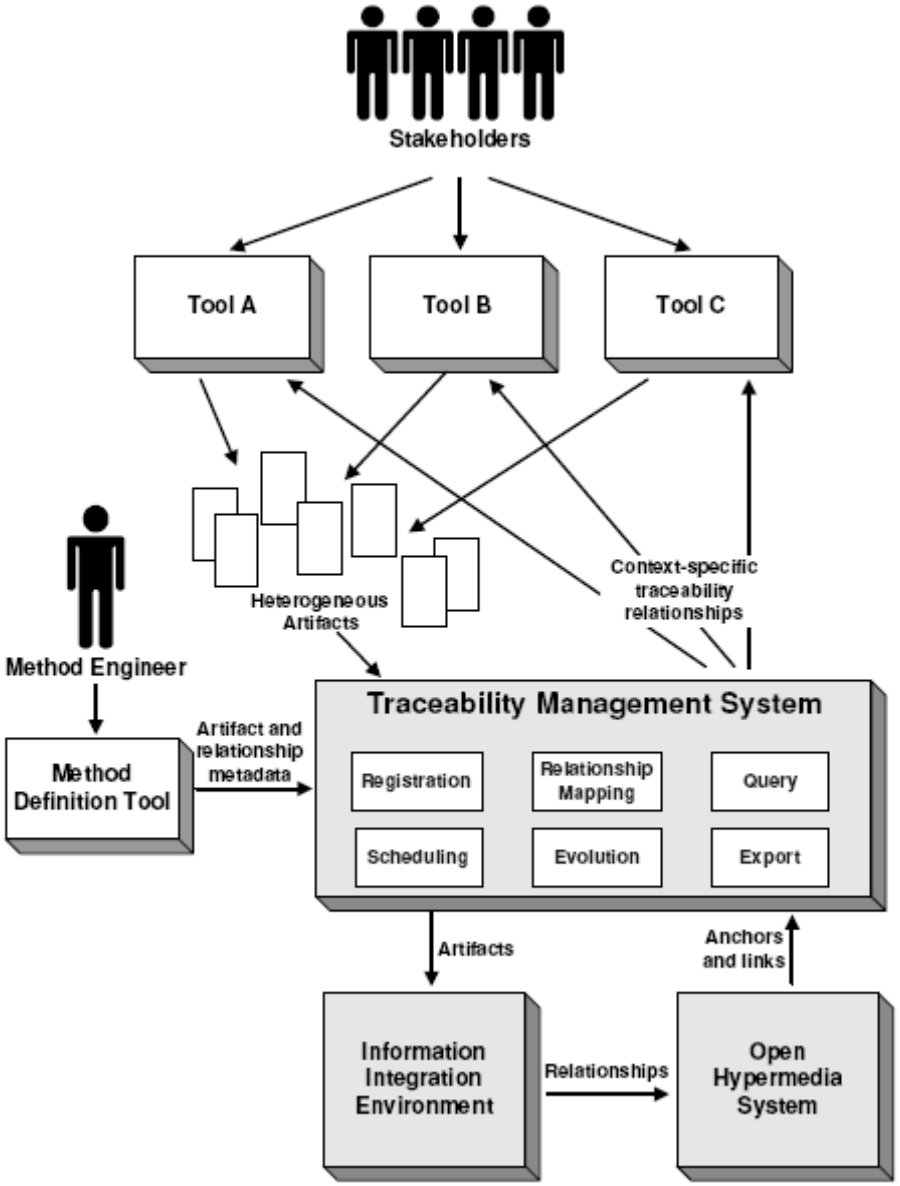

Figura 3.1 Fluxo de atividades do TraceM

Os *stakeholders* criam os artefatos em ferramentas externas e o *method engineer* é responsável por determinar os metadados dos artefatos e suas relações, configurando quais *Translators* e *Integrators* serão utilizados durante a importação, este ator também define quais tipos de artefatos e relacionamentos são interessantes de serem descobertos no momento.

Após a importação dos metadados dos artefatos, os *Integrators* fazem buscas automáticas por relacionamentos. Depois, os relacionamentos

descobertos são repassados para o sistema *Open Hypermedia* que gera um relatório HTML com os links entre os artefatos e também exporta os relacionamentos para as ferramentas onde os artefatos foram originalmente criados.

Existem 6 serviços básicos oferecidos pelo TraceM:

- *Cadastro:* é possível inserir novos *Integrators* e *Translators,* bem como novos artefatos e tipos de relacionamentos;
- *Agendamento:* possibilidade de agendar atualizações dos artefatos modificados externamente;
- *Evolução:* análise de como os relacionamentos se modificaram ao longo do tempo;
- *Consulta:* customização de filtros dos artefatos e relacionamentos que são mais interessantes para os usuários no momento;
- *Exportação:* gerar relatórios em HTML sobre os relacionamentos encontrados;

- *Criação de novos tipos de relacionamentos*:

É possível criar novos tipos de relacionamentos a partir de uma seqüência de relacionamentos que segue um padrão. Um exemplo: entre um requisito e um trecho de código-fonte existe um relacionamento do tipo “implementado por” e entre esse mesmo trecho de código-fonte e um caso de teste existe um relacionamento do tipo “testado por”, o usuário pode desejar que, para toda cadeia de relacionamentos que sigam este padrão, seja criado automaticamente um novo tipo de relacionamento chamado “validado por” entre o requisito e o caso de teste.

3.2. CaliberRM

O CaliberRM é a ferramenta de gerenciamento de requisitos da Borland [Caliber User Guide]. Este software é capaz de manter o rastreamento entre requisitos criados em um projeto e entre requisitos de projetos distintos. Também é possível manter a rastreabilidade entre seus requisitos e artefatos produzidos externamente nas ferramentas da Borland, como por exemplo, se existe relação com casos de testes criados no TestDirector e arquivos fontes mantidos pelo StarTeam.

Este software também permite visualizar relações indiretas entre requisitos, ou seja, se o requisito R1 se relaciona com o R2 e o R2 se relaciona com o requisito R3, o CaliberRM infere um relacionamento indireto entre R1 e R3.

As relações são bidirecionais, podendo ser do tipo “trace from” e/ou “trace to”, indicando qual requisito causa impacto e qual é o afetado.

Toda a rastreabilidade é mantida manualmente pelo usuário. Para se criar um relacionamento, um requisito é escolhido pelo usuário, em seguida a direção e o segundo requisito (ou artefato externo) são escolhidos. Também é possível modificar e deletar o relacionamento quando desejado.

Quando algum artefato é modificado dentro do CaliberRM, os relacionamentos que partem dele são marcados como “suspeitos”, cabendo ao usuário analisar a mudança que foi feita e decidir se o relacionamento ainda é válido.

Existem duas representações possíveis para visualizar os relacionamentos entre os requisitos: matriz e diagrama.

Diagrama

No modo diagrama, um grafo é mostrado, os nós são os requisitos e as arestas representam os relacionamentos. As direções são indicadas por setas

nas extremidades das arestas. É possível organizar o grafo arrastando os nós para uma melhor visualização.

O diagrama é gerado a partir de um determinado requisito escolhido e nem todos os relacionamentos do projeto em questão são mostrados: apenas são visualizados aqueles requisitos que têm relações diretas e indiretas com o requisito escolhido pelo usuário.

Se um requisito tem relação direta ou indireta com ele mesmo, o diagrama alerta o usuário com um ícone de relação circular.

Abaixo um exemplo de diagrama gerado pelo Caliber:

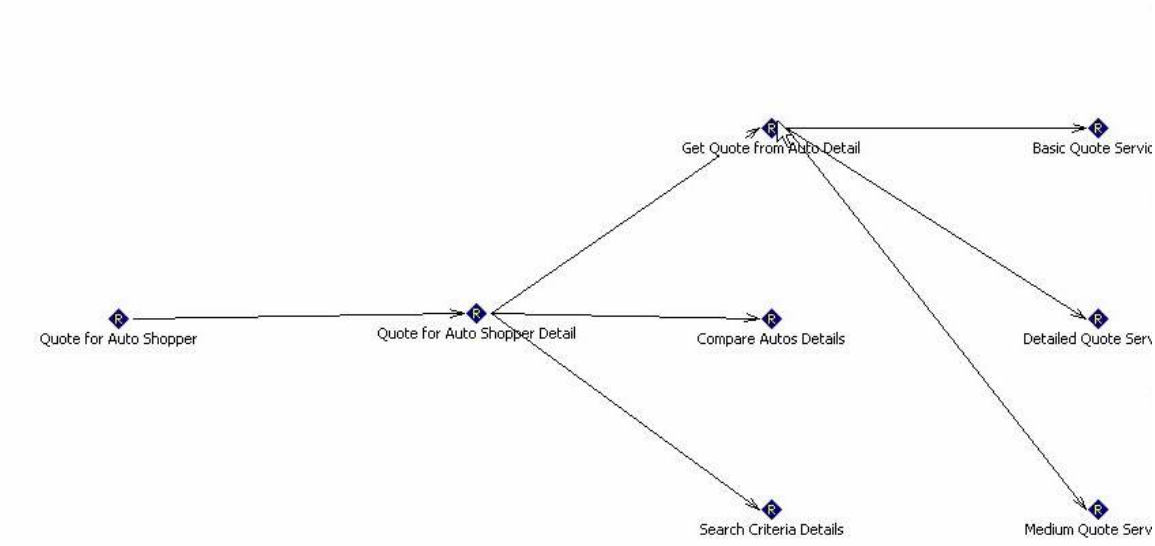

Figura 3.2 Exemplo de diagrama do Caliber

Matriz

No modo matriz, todos os requisitos são listados na primeira linha e na primeira coluna de uma tabela, os relacionamentos são representados por setas direcionais nas células dessa matriz.

É possível também configurar filtros para visualizar na Matriz apenas os dados que interessam ao usuário no momento, podendo-se escolher a natureza dos artefatos (requisito, caso de teste, etc.) e os tipos dos requisitos (requisitos do usuário, requisitos do sistema, etc.). Após a filtragem, o usuário pode salvar

as opções numa visão, podendo utilizá-las posteriormente para gerar novamente a matriz; também é possível gerar relatórios com as informações obtidas.

3.3. Rational RequisitePro

O RequisitePro é a ferramenta de gerenciamento de requisitos da Rational [RequisitePro]. As funcionalidades oferecidas relativas ao controle da rastreabilidade se assemelham às do CaliberRM, porém podemos destacar suas principais características a seguir:

Percepção automática de quebra de relacionamento

A partir dos requisitos criados no RequisitePro é possível gerar documentos Word a partir de templates.

Caso o texto relativo a um requisito dentro do documento Word for modificado pelo usuário, o RequisitePro automaticamente marca como "suspeitos" quaisquer links que este requisito venha a ter com outros.

Cabe ao usuário então, decidir se ainda existe, ou não, relacionamento entre os requisitos dependentes.

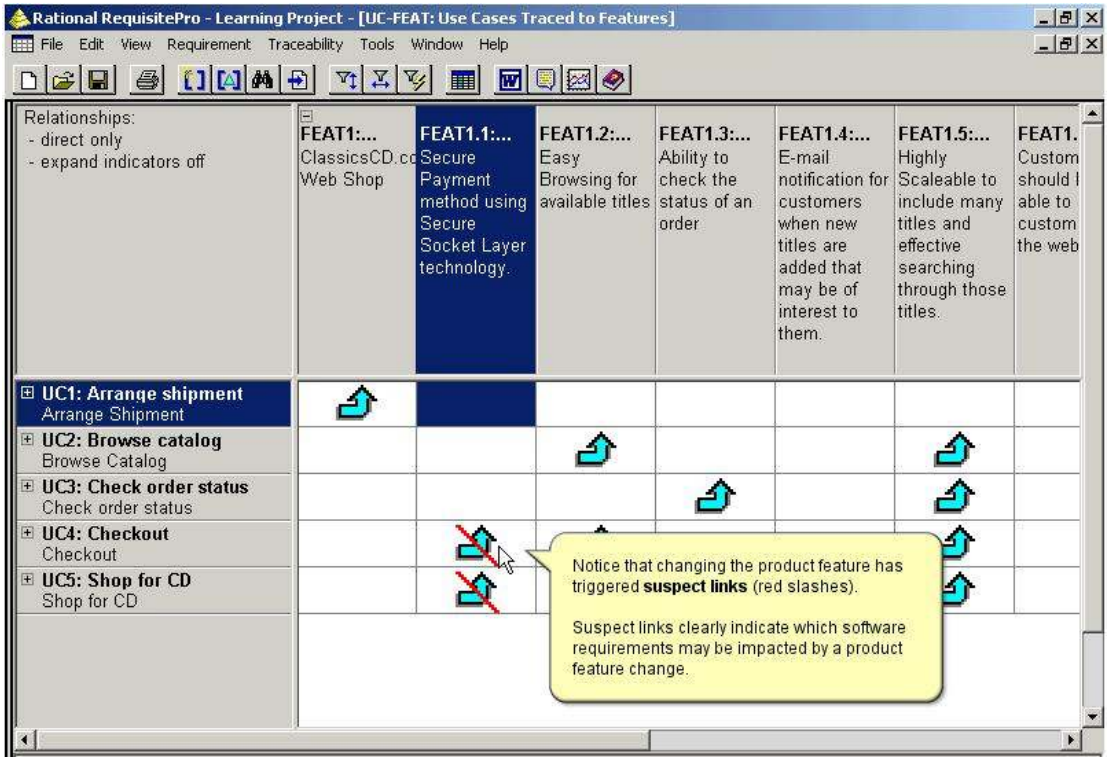

Figura 3.3.1 RequisitePro alertando links suspeitos

Criação de visões da matriz de rastreabilidade

No RequisitePro existe esta funcionalidade que é bastante interessante. É possível construir consultas específicas sobre um conjunto de requisitos, e descobrir, por exemplo, quais funcionalidades do software em questão foram definidas mas ainda não foram implementadas.

É possível salvar uma visão da matriz resultante da consulta, deixando claro para todos envolvidos o que ainda falta ser feito.

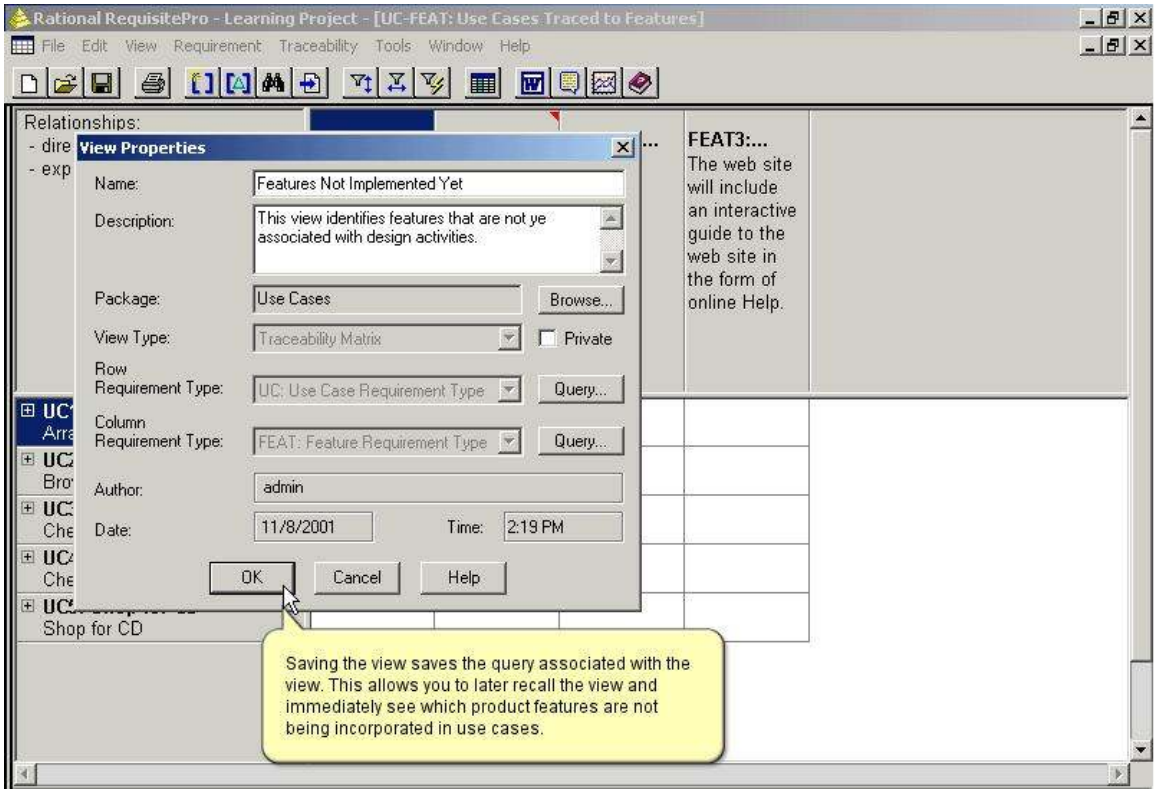

Figura 3.3.2 Criação de visões de matrizes no RequisitePro

3.4. Comparativo entre as ferramentas

Através das descrições acima é possível concluir que o TraceM foi arquitetado para ser bem extensível e o seu único objetivo é prover o gerenciamento da rastreabilidade entre vários tipos de artefatos.

Com outro foco, o CaliberRM e o RequisitoPro, que são bem parecidos no que diz respeito à rastreabilidade, têm como objetivo principal facilitar o gerenciamento de requisitos e neles a rastreabilidade é uma função secundária. Além disso, essas duas ferramentas são capazes de importar apenas artefatos gerados em ferramentas da própria Borland e Rational, o que não permite estender para qualquer tipo de ferramenta.

No entanto, o TraceM não foi implementado e por isso não é utilizado pelas organizações como são as outras duas ferramentas que em geral fazem parte de um pacote de aplicações integradas e são bem populares, oferecendo muitas outras funcionalidades.

Uma funcionalidade bastante interessante que o TraceM oferece é a de ser possível criar novos tipos de relações dando um significado à relação e provendo um entendimento melhor sobre as dependências entre os artefatos. No RequistePro e CaliberRM é possível apenas indicar a direção de dependência entre dois requisitos.

A visualização da rastreabilidade como um todo no TraceM é bastante prejudicada pois são gerados arquivos HTML e o usuário tem que ir navegando para descobrir os relacionamentos, o que não acontece na visualização através de matrizes que oferece uma visão bem mais ampla. Outro modo bastante interesse de visualização, porém menos comum, é o de diagramas que é oferecido apenas pelo CaliberRM: através de grafos podemos focar a atenção apenas em um requisito e analisar onde ele impacta e de onde sofre impacto, além de poder organizá-los espacialmente.

Quanto à automação na descoberta de relações o CaliberRM é muito fraco, ele apenas marca relacionamentos como suspeitos quando um dos dois requisitos é modificado então cabe ao usuário gerenciar toda a rastreabilidade

manualmente. O RequisitePro também oferece a função de marcar automaticamente uma relação como suspeita e ele faz isso melhor pois também é capaz de identificar mudanças ocorridas em documentos de requisitos em arquivos Word. O TraceM é o melhor nesse aspecto porque a maioria das relações são descobertas automaticamente pelos *Integrators* e o usuário precisa apenas verificar e confirmar ou não as descobertas.

As três ferramentas permitem o acompanhamento da evolução da rastreabilidade. Apenas o RequisitePro e o CaliberRM permitem ao usuário ter diversas visões sobre a rastreabilidade e salvá-las.

Abaixo segue o resumo comparativo das três ferramentas numa matriz:

	TraceM	**RequisitePro**	**CaliberRM**
Popularidade	Baixa	Alta	Alta
Automação	Alta	Média	Baixa
Extensibilidade	Alta	Baixa	Baixa
Visualização com matrizes	Não	Sim	Sim
Visualização com grafos	Não	Não	Sim
Tipificação de relações	Sim	Não	Não
Acompanhamento no tempo da rastreabilidade	Sim	Sim	Sim
Permite criar visões sobre as relações	Não	Sim	Sim

Tabela 1 Tabela comparativa das ferramentas

4. Descrição do Trace Tracker

Após o comparativo das ferramentas no capítulo anterior foram elicitados os principais requisitos do protótipo proposto por este trabalho.

Foram aproveitadas as principais características do TraceM porque esta se mostrou a melhor ferramenta com exceção da parte de visualização que é bastante pobre (apenas HTML). Por este motivo foram aproveitados os modos de visualização de matriz existente tanto no CaliberRM como no RequisitePro e a opção de filtros customizados pelo usuário, melhorando assim o entendimento da rastreabilidade.

Além disso, Trace Tracker chegou a ser implementado e está disponível publicamente, ao contrário do TraceM, que nem é possível saber se seu protótipo foi ou não implementado.

O Trace Tracker foi feito em Java e utiliza PostgreSQL como banco de dados. Assim como o TraceM, o sistema foi feito de tal modo que permita importar artefatos e relacionamentos de qualquer ferramenta externa desde que seja construído um Integrador para tanto. Os artefatos em si não são importados, apenas referências para eles ficam no banco de dados do Trace Tracker.

Logo após o documento de requisitos estão o diagrama de classes e as funcionalidades que realmente foram implementadas pelo sistema e suas telas.

4.1. Requisitos Funcionais

[RF-001] Manter cadastro das relações

Descrição:

O usuário poderá incluir, modificar e excluir os relacionamentos existentes entre os artefatos. Todas essas ações devem ser feitas a partir da matriz de rastreabilidade.

Quando for modificar uma relação, o usuário poderá modificar suas propriedades, tais como: direção, tipo e se a relação é suspeita ou não. Também será possível visualizar se a relação foi descoberta automaticamente ou não, sua data de criação e de qual software foi importada.

Prioridade: Essencial.

Pré-condições:

- Um projeto deve estar aberto.
- A matriz de rastreabilidade deve estar aberta.

[RF-002] Manter projetos

Descrição: O usuário poderá incluir, modificar e excluir projetos. Um projeto é a um agrupamento de artefatos e suas relações, um projeto é a entidade base do sistema e cada um tem um nome que o identifica.

Prioridade: Essencial.

Pré-condições: Nenhuma

[RF-003] Abrir projeto

Descrição: O usuário poderá escolher um dos projetos cadastrados e abri-lo.

Prioridade: Essencial.

Pré-condições:

- Projeto para ser aberto deve estar cadastrado no sistema.

[RF-004] Visualizar matriz de rastreabilidade

Descrição:

O usuário poderá visualizar a rastreabilidade dos artefatos através de uma matriz. Uma matriz consiste numa tabela onde os artefatos relacionados ficam na primeira coluna e na primeira linha.

Na intersecção das colunas e linhas haverá uma imagem de uma seta indicando a direção da relação (ou seja, da linha para coluna, da coluna para a linha ou em ambas as direções), além de uma indicação se a relação é suspeita ou não. A partir dessa intersecção é que o usuário poderá manter os relacionamentos como descrito em [RF-001].

O usuário poderá configurar quais artefatos serão visualizados nas linhas e colunas da matriz, este filtro deverá ser baseado nos tipos dos artefatos.

Prioridade: Essencial.

Pré-condições:

- Um projeto deve estar aberto.

[RF-005] Importar artefatos

Descrição:

O usuário poderá importar relacionamentos entre artefatos de ferramentas externas. Será possível escolher o Integrador a ser utilizado e configurá-lo no momento da importação.

O Integrador importará apenas os metadados dos artefatos e não os artefatos em si. Relacionamentos poderão ser automaticamente descobertos pelos Integradores e importados para o sistema.

Será possível configurar se o Integrador deverá sobrescrever ou não artefatos e relacionamentos já existentes no sistema.

Prioridade: Essencial.

Pré-condições:

- Um projeto deve estar aberto.
- O Integrador escolhido deve estar cadastrado no sistema

[RF-006] Manter integradores

Descrição: O usuário poderá incluir e excluir os Integradores que poderão ser utilizados por todo sistema.

Prioridade: Essencial.

Pré-condições: Nenhuma.

[RF-007] Agendamento de atualização

Descrição: O usuário poderá agendar atualizações da rastreabilidade de um determinado projeto, configurando os intervalos de tempo e quais Integradores deverão ser utilizados.

Prioridade: Desejável.

Pré-condições:

- Um projeto deve estar aberto.
- Os Integradores escolhidos devem estar cadastrados no sistema.

[RF-008] Controlar evolução da rastreabilidade

Descrição: O sistema deverá ser capaz de controlar as versões dos relacionamentos à medida que os artefatos e relacionamento vão evoluindo. Ao usuário será permitida apenas a visualização das várias versões da rastreabilidade.

Prioridade: Importante.

Pré-condições: Nenhuma.

[RF-009] Exportar rastreabilidade

Descrição: O usuário poderá exportar um projeto para formato XML bem como gerar relatórios em formato HTML e PDF. Todas as informações sobre o projeto em questão deverão ser incluídas, tais como artefatos e relacionamentos e suas propriedades.

Prioridade: Desejável.

Pré-condições:

- Um projeto deve estar aberto.

4.2. Requisitos não-funcionais

[RNF-001] Permitir inclusão de novos Integradores

Descrição: O sistema deve ser construído de tal forma que garanta sua extensibilidade para novos Integradores que venham a ser criados por outros desenvolvedores. Dessa forma será possível importar artefatos das mais diversas ferramentas externas à medida que vão sendo feitos novos Integradores.
Prioridade: Essencial.
Pré-condições: Nenhuma.

4.3. Diagramas de classe

Segue abaixo os diagramas de classe dos principais pacotes do protótipo. Com o intuito de simplificar a visualização dos diagramas algumas medidas foram tomadas, como por exemplo: não foram explicitados os métodos getters e setters das classes; apenas os métodos mais importantes aparecem (não foram mostrados os métodos da classe Repositório por exemplo) e o pacote responsável pela renderização do sistema (swing) também não foi detalhado pois não é importante para o entendimento do sistema como um todo e além disso haverá um capítulo com imagens do protótipo funcionando.

Pacote integração

Neste pacote estão as classes que implementam o Requisito não-funcional [RNF-001]. Ou seja, são as classes responsáveis pela integração com ferramentas externas e que importam artefatos e relacionamentos para o Trace Tracker.

A interface Integrador é o contrato que existe entre o Trace Tracker e os demais Integradores que venham a ser escritos no futuro. Ele obriga os Integradores a implementarem os seguintes métodos:

- init() – para que haja inicialização dos recursos que serão utilizados durante a importação.

- run() - nesse método é onde ocorre a importação propriamente dita, ela deve retornar uma coleção de Relacionamentos entre Artefatos e então o Trace Tracker guarda as informações em seu banco de dados.

- destroy() – para que os recursos inicializados sejam destruídos depois da importação.

- getName() – para que o Trace Tracker mostre um nome amigável para o Integrador na interface gráfica com o usuário.

- setProperties(Properties props) – este método é responsável para configurar os Integradores que necessitam de alguns dados para poderem rodar.

A classe GerenciadorIntegradores é responsável por todo ciclo de vida dos Integradores. Ou seja, é ela quem mostra quais Integradores estão disponíveis no Trace Tracker além de instanciar, configurar, rodar e destruir os Integradores (chamando os métodos init, setProperties, run e destroy nesta ordem).

O GerenciadorIntegradores utiliza técnicas do Java Reflection para instanciar dinamicamente os Integradores e chamar seus métodos.

Para incluir um Integrador no Trace Tracker basta o usuário configurar um arquivo texto e digitar o nome completo de uma classe que implementa a interface Integrador, esta classe deve estar na classpath da aplicação.

Por fim, foi implementado um Integrador de exemplo para testes. Ele simplesmente lê um arquivo texto com alguns requisitos e tenta achar relações entre esses requisitos.

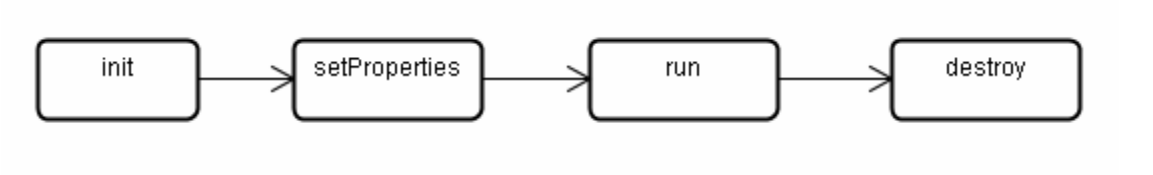

Figura 4.3.1 Ciclo de vida de um Integrador

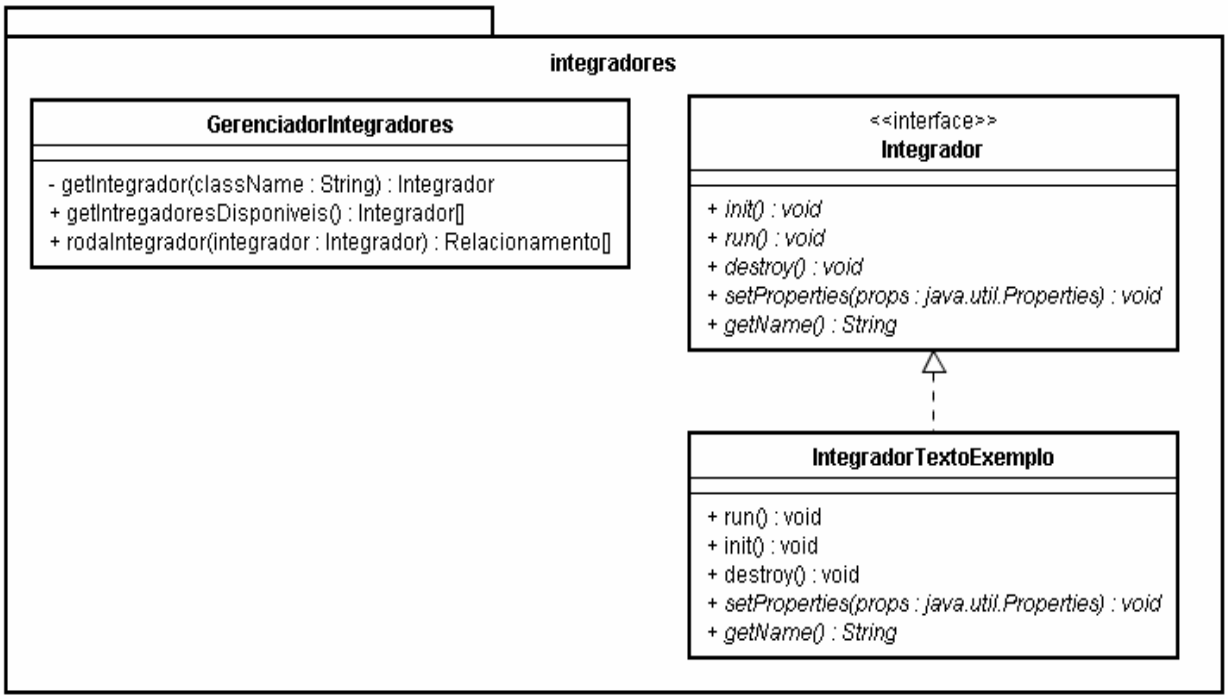

Figura 4.3.2 Digramas de classe do pacote integradores

Pacote de classes básicas

Nesse pacote está o modelo do sistema, todas as classes contidas são persistidas no banco de dados do Trace Tracker com exceção apenas da classe Matriz que serve como ponte dos Relacionamentos e Artefatos para a visualização na interface gráfica do software.

Projeto é a classe principal do sistema, é um conjunto de artefatos e seus relacionamentos.

Um Relacionamento contém dois Artefatos: um é a causa da relação e outro é o alvo. Além disso, um Relacionamento tem uma Direção, indicando se a relação está partindo do artefato1 para o artefato2 (ou seja, se uma mudança no artefato1 impacta no artefato2) ou o contrário ou ainda se o impacto ocorre em ambas direções.

O Artefato tem um tipo (por exemplo, se é um requisito, caso de teste ou código fonte), tem um inteiro que o identifica unicamente e a data de quando foi importado de outro sistema por um Integrador.

Como foi dito, a classe Matriz tem o único propósito de representar os Artefatos e seus Relacionamentos numa matriz. Ela possui os artefatos que serão mostrados na primeira coluna e na primeira linha da matriz e contém os relacionamentos que ficarão representados na intersecção da linha e coluna.

A seguir está o diagrama de classes deste pacote.

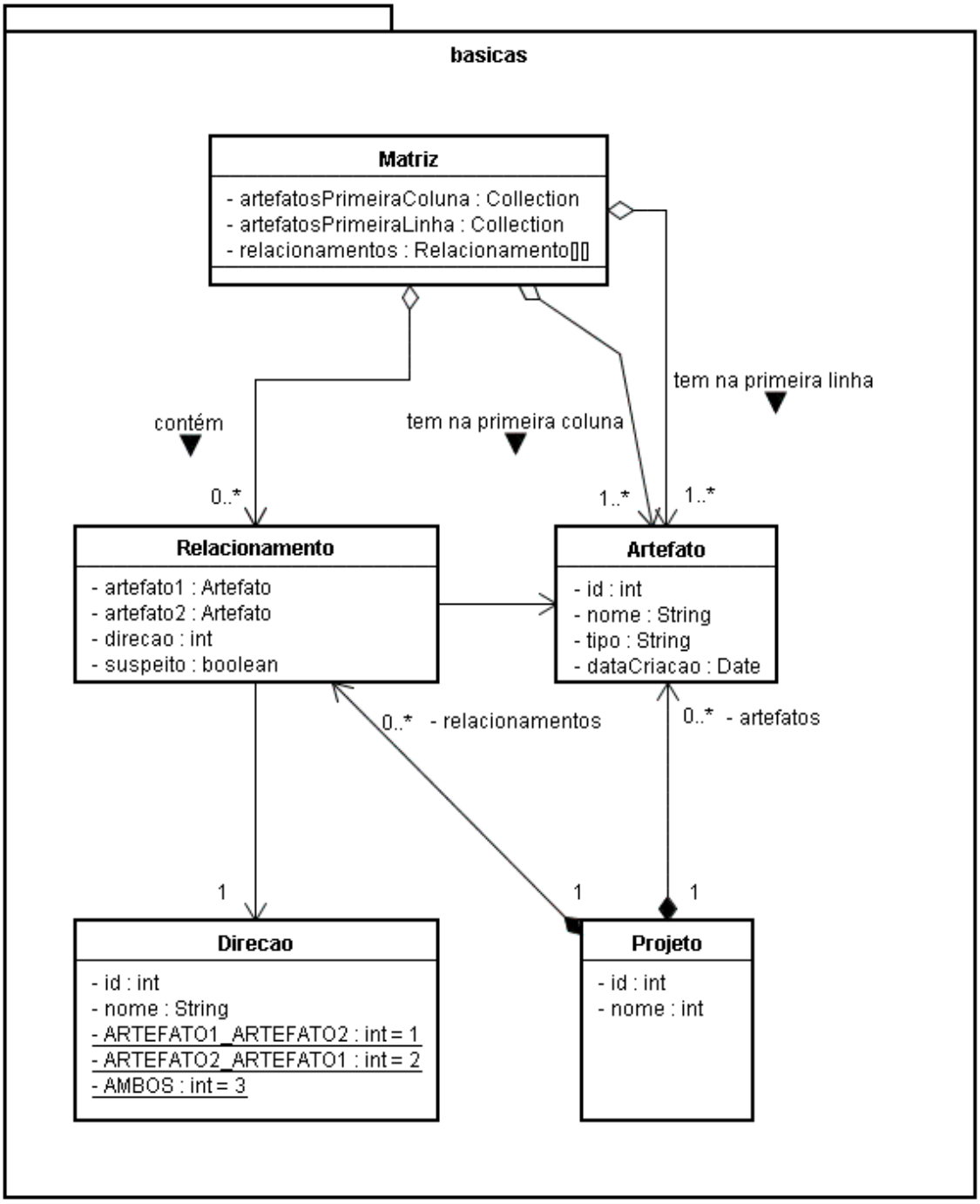

Figura 4.3.3 Digramas de classe do pacote básicas

Pacote de persistência

Este pacote contém apenas a classe Repositório que é responsável por fazer a conexão com o banco e todos os acessos a base do Trace Tracker.

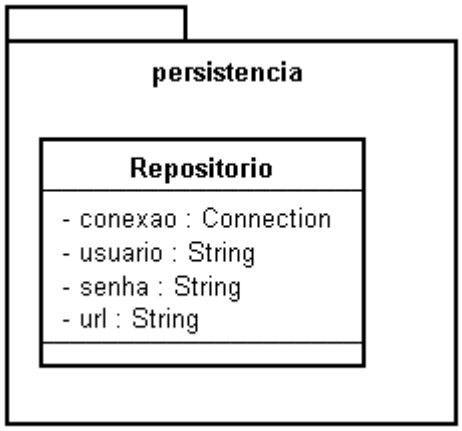

Figura 4.3.4 Diagramas de classes do pacote persistência

4.4. Funcionalidade implementadas no Trace Tracker

Após finalização da modelagem do Trace Tracker veio fase a implementação. Nesta seção serão listadas os casos de usos que foram implementados.

TELA PRINCIPAL

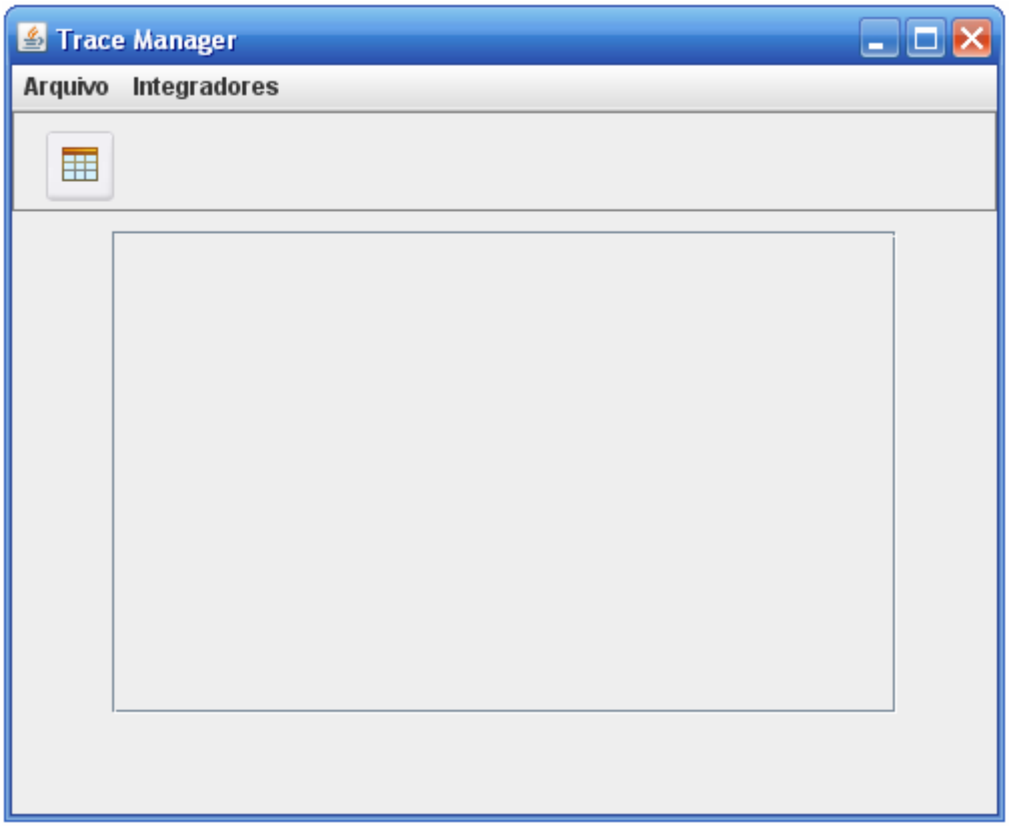

Figura 4.4.1 Tela de abertura do Trace Tracker

MANTER PROJETOS [RF-002]

Através do menu Arquivo o usuário pode criar, editar, abrir e excluir os projetos.

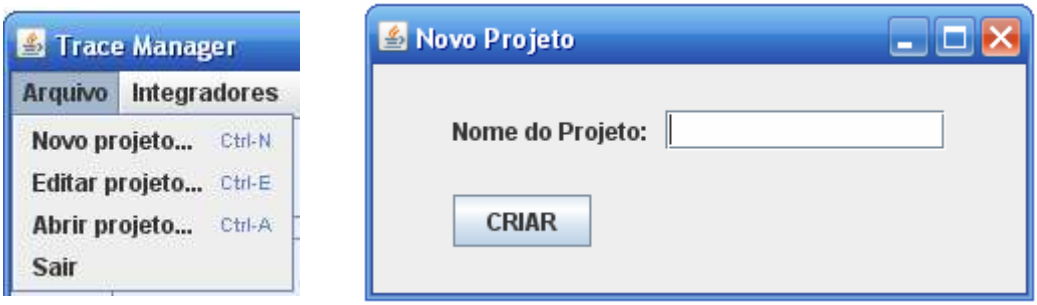

Figura 4.4.2 Tela de novo projeto

ABRIR PROJETO [RF-003]

O usuário deve sempre abrir um projeto pra poder acessar as funcionalidades do sistema.

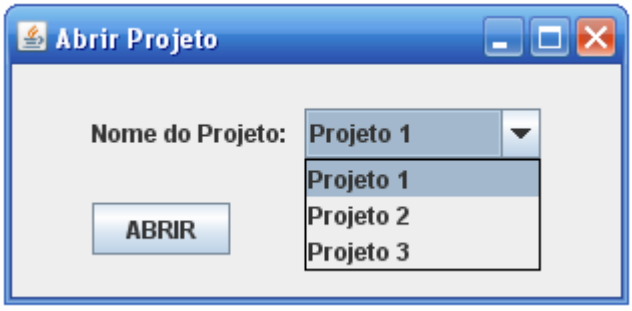

Figura 4.4.3 Tela de abrir projeto

VISUALIZAR MATRIZ DE RASTREABILIDADE [RF-004]

O usuário deve sempre abrir um projeto pra poder acessar as funcionalidades do sistema.

Após clicar no ícone Visualizar Matriz o usuário deve escolher quais tipos de artefato ficarão na primeira linha e na primeira coluna.

Após clicar no botão “visualizar” a matriz de rastreabilidade é aberta. As setas indicam a direção da relação, se é do artefato na linha vertical para o artefato na linha horizontal ou inverso ou se é em ambas as direções.

Com um duplo clique na setas o usuário é levado para a tela de manutenção de relação, descrita a seguir.

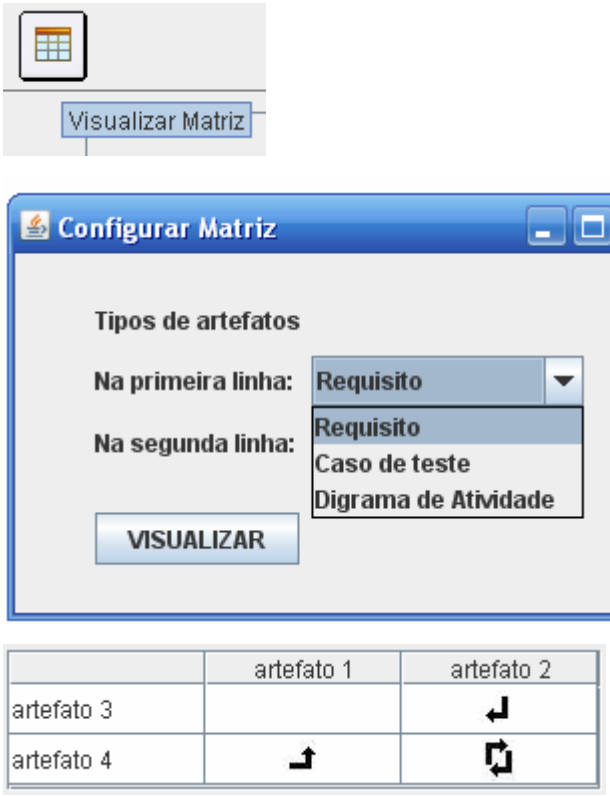

Figura 4.4.4 Tela de visualização da matriz

MANTER RELAÇÃO ENTRE ARTEFATOS [RF-001]

Após selecionar uma das intersecções da matriz o usuário pode incluir, modificar e deletar a relação entre o artefato da coluna e o da linha.

Para deletar, basta escolher a seta vazia. Para incluir ou modificar, basta selecionar umas das setas.

O usuário pode ainda marcar ou desmarcar se a relação ainda é duvidosa ou não no campo “suspeito”.

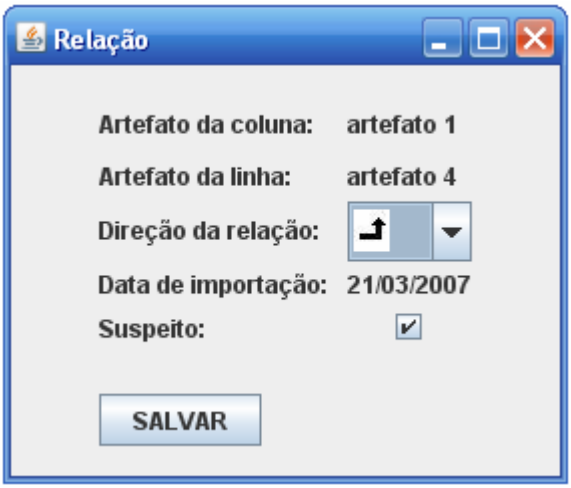

Figura 4.4.5 Manter relação entre dois artefatos

IMPORTAR ARTEFATOS [RF-005]

O usuário pode importar novos artefatos e relacionamentos para o projeto aberto, para tanto basta clicar no menu em “Importar Artefatos”. O usuário então seleciona qual o Integrador que deseja usa no momento.

Após clicar no botão “importar” o Integrador selecionado vai rodar, trazer importar os artefatos e seus relacionamentos, quando concluir a matriz de rastreabilidade aparece atualizada com os novos itens importados.

Figura 4.4.6 Importar artefatos

5. Conclusões

5.1. Considerações finais

Neste trabalho foram estudados os benefícios e principais problemas encontrados durante a manutenção da rastreabilidade entre artefatos produtos do desenvolvimento de softwares.

Um dos principais problemas existentes nessa área é a incapacidade de se relacionar artefatos distintos produzidos por ferramentas distintas. Foram analisados 3 softwares que ajudam no gerenciamento da rastreabilidade, no sentido de integração o TraceM se mostrou o melhor, porém a visualização da rastreabilidade fica bastante prejudicada em formato HTML, o CaliberRM e o RequisitePro se mostraram melhores nessa característica, oferecendo visualizações em grafos e matrizes. O Trace Tracker reuniu as características boas das três ferramentas: oferece importação bem extensível e uma boa visualização da rastreabilidade.

Outro problema relatado neste trabalho foi a falta de centralização da rastreabilidade, cada profissional guarda em geral suas anotações em lugares distintos. Com o Trace Tracker é possível manter a rastreabilidade em lugar só e de forma padronizada, isto facilita sua manutenção e o conhecimento da organização não se perde.

A automação também ajuda bastante no trabalho de manter a rastreabilidade, fica a cargo dos Integradores vasculharem os relacionamentos existentes, bastando ao usuário confirmar ou não. Marcar relacionamentos como “suspeitos” automaticamente também ajuda quando os artefatos vão sendo modificados à medida que o desenvolvimento do software vai evoluindo.

5.2. Dificuldades encontradas

Uma das dificuldades encontradas na implementação do Trace Tracker foi a seguinte: já que fica a cargo dos Integradores definirem o tipo da relação descoberta, como podemos padronizar estes tipos? Um desenvolvedor pode construir um Integrador que considera um tipo de relacionamento como sendo “satisfaz” e outro desenvolvedor pode chamar este mesmo tipo de “sastifies”. Uma solução seria o Tracer Tracker deixar a cargo do usuário escolher o tipo da relação manualmente.

Muitas vezes os artefatos são textuais e não existe uma maneira de identificá-los unicamente. Existe um requisito elicitado para o Trace Tracker que é o de configurar o Integrador e escolher se os artefatos já existentes na base de dados serão ou não sobrescritos, porém fica impossível verificar a existência de um artefato se não existe identificador para o mesmo.

A falta de um estudo de caso dificulta uma análise melhor da eficácia do Trace Tracker.

5.3. Trabalhos futuros

Alguns requisitos não essenciais não foram implementados. Estes tornariam o Trace Tracker mais interessante: agendamento para atualizações automáticas, acompanhamento da evolução da rastreabilidade e exportação da rastreabilidade para formatos diversos.

Um estudo de caso utilizando o Trace Tracker também é necessário.

Referências

IEEE STD 610.12-1990, “IEEE Standard Glossary of Software Engineering Terminology,” IEEE, New York (September 1990).

Artifact definition, Wikipedia
http://en.wikipedia.org/wiki/Artifact_%28software_engineering%29

Model Traceability, IBM Systems Journal, Volume 45, Number 3, 2006.
http://www.research.ibm.com/journal/sj/453/aizenbud.html

O. C. Z. Gotel and A. C. W. Finkelstein, “An Analysis of the Requirements Traceability Problem,” Proceedings of the First International Conference on Requirements Engineering, Utrecht, The Netherlands (1994), pp. 94–101.

S. A. Sherba, K. M. Anderson, and M. Faisal, “A Framework for Mapping Traceability Relationships,” Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering, Montreal, Canada (September 2003).

K. Østerbye and U. K. Wiil. The Flag Taxonomy of Open Hypermedia Systems. In Proceeding of the Seventh ACM Conference on Hypertext, pages 129–139, Washington, DC, USA, 1996.

K. M. Anderson, S. A. Sherba, and W. V. Lepthien. Towards Large-Scale Information Integration. In Proceedings of the 24th International Conference on Software Engineering, pages 524-535, Orlando, FL, USA, May 2002.

Caliber 2005 User Guide, capítulo 16 - Traceability

P. Arkley, P. Mason, and S. Riddle, “Position Paper: Enabling Traceability,” *Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering,* Edinburgh, Scotland (September 2002), pp. 61–65.

RequisitePro Product Library
http://www-306.ibm.com/software/awdtools/resources/reqpro.html?S_CMP=rnav

DOORS
http://www.telelogic.com/products/doors/index.cfm

B. Ramesh and M. Jarke, “Toward Reference Models for Requirements Traceability,” *IEEE Transactions on Software Engineering* **27**, No. 1, 58–93 (January 2001).

A. T. T. Ying, G. C. Murphy, R. Ng, and M. C. Chu-Carroll, “Predicting Source Code Changes by Mining Change History,” IEEE Transactions on Software Engineering 30, No. 9, 574–586 (September 2004).

Orientador

Prof. Ph.D. Alexandre Marcos Lins de Vasconcelos

Aluno

Bruno Farache