# Uma abordagem semi-automática para a manutenção de links de rastreabilidade

Marcelio Leal, Mayara Figueiredo e Cleidson R. B. de Souza.
*Faculdade de Computação*
*UFPA - Universidade Federal do Pará*
*marcelioleal@ gmail.com, {mayara,cdesouza}@ufpa.br*

## Resumo

*Atualmente, uma grande parte das falhas que acontecem em projetos de software está relacionada aos requisitos, tanto na sua elicitação quanto na sua evolução. Assim, a gerência de requisitos adquire um papel muito importante em um projeto, visto que através dela é possível ter maior controle sobre os requisitos. Nesse sentido, a importância da evolução, manutenção e da validade dos requisitos do projeto é um fator decisivo para este controle. É neste momento que a rastreabilidade se faz útil. Através dela é possível estabelecer ligações entre os requisitos e os diversos artefatos produzidos, acompanhando assim a evolução do projeto. A proposta deste trabalho é apresentar uma abordagem semi-automática para a implementação da rastreabilidade de requisitos, descrevendo uma maneira de efetuar esta rastreabilidade de forma mais prática e que possa ser utilizada em diversos tipos de projetos de software.*

## 1. Introdução

Visando manter a qualidade do software produzido e a satisfação do cliente, a engenharia de software tradicionalmente enfatiza uma responsabilidade em elicitar os requisitos e garantir que eles evoluam corretamente no decorrer do projeto. A gerência de requisitos é a área do processo que tem como objetivo controlar a criação e evolução destes requisitos [25]. As principais atividades da gerência de requisitos segundo Kotonya & Sommerville [24] são: gerenciar os requisitos e suas mudanças e gerenciar os links entre os requisitos, e entre os artefatos da fase de requisitos e os demais artefatos produzidos ao longo do processo de desenvolvimento de software.

Neste contexto, percebe-se uma grande preocupação em se ter uma visão dos relacionamentos que envolvem os requisitos. De fato, o sucesso da gerência de requisitos depende diretamente de quão bem definidos são estes relacionamentos entre os requisitos e outros conjuntos de artefatos gerados durante o processo de desenvolvimento [23].

A criação desses relacionamentos, que são chamados *links de rastreabilidade*, entre os itens é responsabilidade da Rastreabilidade de Software (RS). A RS é a capacidade de relacionar artefatos criados durante o processo de desenvolvimento de software, relacionar os *stakeholders* que contribuíram com o desenvolvimento de um artefato, e as decisões de projeto que contribuíram para a construção dos artefatos [08]. Estes relacionamentos descrevem o sistema sob diferentes perspectivas ou níveis de abstração [08].

A RS tem se mostrado um importante aspecto de apoio a todas as fases do processo de desenvolvimento de software, além de contribuir para uma melhor qualidade do produto final [08, 09, 13, 14]. Apesar destes benefícios claros, a gerência de requisitos, especialmente a rastreabilidade de requisitos, ainda é uma atividade cara [23]. Este custo depende diretamente do nível de detalhe das atividades, da diversidade das informações coletadas e, principalmente, de como é feito o processo de criação das informações da rastreabilidade.

O objetivo deste trabalho é descrever uma abordagem semi-automática para a criação e manutenção da RS a partir dos artefatos produzidos durante o processo. Esta abordagem visa maximizar a qualidade dos links e itens de rastreabilidade encontrados, utilizando como base: a combinação de metodologias de processo de desenvolvimento de software; Unifield Modeling Language (UML); *templates* de artefatos; análise de documentos textuais; e a definição de regras para busca dos itens e links de rastreabilidade. A utilização de cada um destes aspectos da abordagem traz benefícios ao processo de rastreabilidade, tal que esta combinação agrega valor as informações extraídas gerando resultados com bem menos esforço aplicado a esta tarefa. Por exemplo, a utilização de templates possibilita melhores resultados na análise de informações não-estruturadas, já que os

templates estruturam as informações em áreas pré-definidas, possibilitando uma busca mais eficiente pelos itens e links de rastreabilidade [29].

O restante deste trabalho está organizado como segue. Na Seção 2 é apresentada a motivação e os trabalhos relacionados. Na Seção 3, é apresentado o detalhamento da abordagem, enquanto que a Seção 4 traz o detalhamento da arquitetura proposta. A Seção 5 apresenta a ferramenta desenvolvida utilizando a abordagem proposta. Finalmente, conclusões e trabalhos futuros são apresentados na Seção 6.

## 2. Motivação – Contexto e Métodos atuais

### 2.1. Rastreabilidade de Software

Segundo El Emam [26], a falta de habilidade para manter o documento de requisitos consistente é um dos principais problemas relacionados à gerência de requisitos. Isto acontece pela natureza instável dos mesmos. Essa instabilidade dos requisitos é um fator decisivo para os riscos de pressão excessiva do cronograma, cerca de 50% de falhas detectadas na fase de testes, e a não aceitação do produto final [27, 28]. Estes problemas poderiam ser evitados em fases iniciais se houvesse uma visão geral da evolução e implementação de cada requisito. Assim, torna-se necessária a implementação de um processo de rastreabilidade de software (RS).

A RS é uma atividade importante que propicia, entre outras vantagens, uma análise da completude e corretude do software acompanhada a partir dos requisitos [08], uma análise de riscos e impactos sobre modificações/integrações no software [08], a identificação de possíveis conflitos entre os requisitos [08, 19, 20], um aumento da reutilização a partir da identificação de componentes utilizando as características descritas nos requisitos e os links entre os mesmos [08] e testes contínuos sobre os requisitos durante todo o ciclo de desenvolvimento [21]. A rastreabilidade também permite que possíveis erros sejam detectados de maneira antecipada e corrigidos em fases ou iterações iniciais [09], isto gera uma economia de recursos considerável, já que o custo das modificações aumenta consideravelmente quanto mais próximas elas sejam das fases de implementação e manutenção [30].

A RS também permite a realização de análise de riscos que consiste em modificar os riscos e o impacto gerado a partir de modificações nos requisitos. Esta análise identifica quais são os possíveis artefatos e requisitos impactados pelas modificações e, por conseqüência, auxilia no cálculo do esforço que irá se levar para implementar essa modificação. Estes pontos já apresentam benefícios para justificar os trabalhos na área da rastreabilidade [09, 22].

Uma das classificações da rastreabilidade se divide em pré-rastreabilidade e pós-rastreabilidade. A primeira conecta os artefatos utilizados para a criação dos requisitos (ou parte deles) com os requisitos criados, enquanto a segunda relaciona os requisitos com eles mesmos, com outros artefatos ou parte deles que os sucedem nas demais fases do processo de desenvolvimento de software [31]. Ambas abordagens proporcionam visões importantes para os desenvolvedores e gerentes do projeto.

Apesar de todas essas vantagens e de anos de pesquisa na área, estudos empíricos das necessidades e práticas de rastreabilidade na indústria apresentam um cenário onde a rastreabilidade é raramente estabelecida em ambientes industriais [10, 11]. Segundo Spanoudakis [08], isto se deve principalmente pela dificuldade de se automatizar a geração dos itens e links de rastreabilidade com a clareza e semântica necessárias para assim prover efetivamente os benefícios já citados com um bom custo-benefício.

Neste contexto, as principais áreas de estudo da rastreabilidade são: a geração (semi)automática de rastreabilidade e a atribuição de mais sentidos semânticos aos links de rastreabilidade [08, 13]. A primeira área de estudo visa diminuir o custo da utilização da rastreabilidade em um processo de desenvolvimento de software, enquanto que a segunda área adiciona visões para as diferentes partes interessadas durante todo o processo, possibilitando assim um aumento de qualidade do processo inteiro.

Em geral, os tipos de links de rastreabilidade pertencem a oito principais grupos: evolução, dependência, sobreposição, satisfacibilidade, abstração, conflito e *rationale* entre os itens de rastreabilidade, ou contribuição entre os stakeholders e os itens de rastreabilidade [08, 10]. Cada tipo de link disponibiliza uma ou mais visões sobre o produto ou o processo.

A atribuição de mais sentidos semânticos aos links de rastreabilidade apresenta dois benefícios diretos: uma documentação mais clara e consistente do sistema e um processo de manutenção menos dependente de especialistas [09].

### 2.2. Construção dos Links

Atualmente, as abordagens utilizadas para rastreabilidade seguem três linhas principais: construção manual de links de rastreabilidade, foco nos dados extraídos do controle de versão para aferir informações de rastreabilidade e uma visão orientada a eventos.

Os autores [32, 33, 34] que utilizam a criação manual da rastreabilidade defendem que somente este

tipo de criação traz a riqueza de informações necessária para utilização do processo de rastreabilidade. De fato, alguns tipos de links têm característica de criação manual pela sua natureza subjetiva, porém este tipo de criação é passível de erro, consome tempo, e é complexo [08]. Assim, apesar das vantagens descritas, raramente se terá um processo, na indústria, estabelecido manualmente [08].

Outros autores utilizam a geração semi-automática de links que, em geral, pode ser de dois tipos: baseadas em links pré-definidos e as baseadas em processos de software [08]. Algumas das principais abordagens são [14, 15, 16, 17, 18, 19]. As baseadas em links pré-definidos requerem que os usuários definam previamente alguns links, que servirão como base para a definição dos demais. As baseadas em processos seguem as especificidades das fases e características de um processo de desenvolvimento de software, como os modelos dos artefatos, fases, atividades, entre outros e utilizam estas informações para criar os links.

Entretanto, mesmo com a vantagem de melhorar a automatização e qualidade, as abordagens semi-automáticas ainda não conseguiram a qualidade e a precisão necessária para serem adotadas em grande escala pela indústria [08].

A geração automática de links tem se baseado principalmente nas abordagens de recuperação de informações, regras de rastreabilidade e axiomas de inferência. Porém, os resultados da geração automática ainda não são satisfatórios em todos os tipos de links de rastreabilidade. Algumas destas abordagens melhoram a geração dos links, porém geram mais links que não são reais, apresentam também problemas de performance [08]. Alguns protótipos e ferramentas já foram implementadas baseadas nestas abordagens, porém ainda não estão em um nível de maturidade suficiente para uma adoção em larga-escala [08].

A abordagem proposta neste trabalho reúne algumas das características propostas em outros trabalhos que visam a geração (semi)automática como as regras de rastreabilidade [16] e o foco no processo, porém com flexibilidade suficiente para incorporar diversos tipos de processos, dos mais simples até instâncias mais formais.

## 3. Descrição da Abordagem

Em um processo de desenvolvimento de software temos, simplificadamente, dois tipos de artefatos: artefatos que possuem uma sintaxe rigorosa e precisa, como: códigos-fonte e diagramas UML; e os artefatos baseados em textos livres, como: especificações de caso de uso e especificações de casos de testes. A descoberta de informações em artefatos com sintaxe precisa é bastante utilizada em várias abordagens/ferramentas, como na suíte Rational [02]. Em geral, os resultados possuem qualidade já que estes artefatos seguem sintaxes precisas e têm atributos semânticos explícitos, o que permite a precisa extração de informações. Em contraste, a extração de informações nos artefatos baseados em textos é uma tarefa mais difícil e cuja qualidade ainda é variável [08], principalmente porque estes artefatos geralmente utilizam linguagem natural. Logo, técnicas de Processamento de Linguagem Natural (PLN) são utilizadas para extrair informações dos artefatos.

Para otimizar a qualidade das informações extraídas dos artefatos produzidos durante o processo de desenvolvimento, a abordagem proposta neste trabalho baseia-se na utilização de:

- *templates* de artefatos, que objetivam fornecer uma estrutura inicial aos documentos, e assim permitir a padronização dos artefatos textuais;
- técnicas de processamento de linguagem natural para extração das informações dos artefatos textuais;
- uma arquitetura em 3 camadas, baseada em componentes e padrões de projeto, para o desenvolvimento de uma solução flexível, reconfigurável e escalável;
- UML e padrões como XMI, que são utilizados largamente na indústria de desenvolvimento de software.

O objetivo final da abordagem proposta é extrair as informações dos artefatos em seus formatos iniciais.

A Figura 1 apresenta uma visão geral sobre o processo de rastreabilidade a partir do artefato, neste caso uma Especificação de Caso de Uso (ECU).

No processo apresentado pela Figura 1, primeiramente são extraídas as informações do artefato inicial. Estas informações podem gerar Itens de rastreabilidade, como um Caso de Uso, ou gerar links entre os Itens, como a informação dos Atores Relacionados com o Caso de Uso. O cenário de links apresenta os itens e seus links. As setas entre as informações extraídas e o cenário de links indicam onde estas informações serão úteis na rastreabilidade.

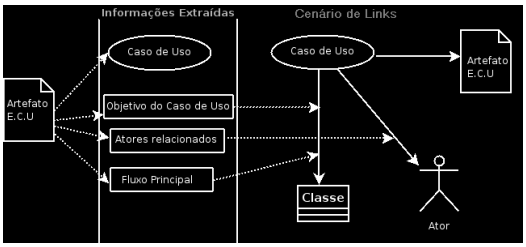


**Figura 1. Visão simplificada do processo de extração de um artefato (ECU).**

Deve-se ressaltar que para a utilização da abordagem não é necessário que todos os artefatos de uma fase do processo sejam criados, mas é necessário que os artefatos baseados em textos, como as especificações de casos de uso, sigam *templates* pré-definidos e, importante, que os artefatos baseados em diagramas sejam baseados em um padrão como o XMI. Estas restrições não são consideradas significativas dada a popularidade e ampla utilização de padrões de processo de software como o Rational Unified Process, que definem *templates* para seus artefatos.

## 3.1. Templates

Um *template* é um arcabouço para um determinado tipo de artefato de software e tem o objetivo de padronizar, guiar e facilitar a sua construção. Templates são utilizados em quase todas as atividades do processo de software, desde o planejamento do projeto, passando pela fase de requisitos, até à implementação. A Figura 2 apresenta um exemplo de *template* do artefato ECU baseado no Rational Unified Process (RUP).

**1.1 Cadastrar Usuário**

**1.1.1. Breve Descrição**

Este caso de uso trata da ação de cadastrar contas de usuários.

**1.1.2. Fluxo Principal**

P1. O Coordenador de Operações aciona a ação Cadastrar Usuário.

P2. Insere as informações do usuário.

P3. O fluxo principal termina com a confirmação do cadastro.

**Figura 2. Template do Artefato Especificação de Caso de Uso.**

Na maioria das metodologias utilizadas por processos ou ferramentas, existe uma quantidade considerável de artefatos que são baseados em texto. Por exemplo, a ECU ou a especificação de casos de testes. Neste caso, os *templates* são utilizados para formatar as áreas editáveis do artefato, guiar o processo de criação e exemplificar cada área do artefato apresentando exemplos e as regras de obrigatoriedade de cada item do artefato.

A abordagem proposta neste artigo utiliza *templates*, pois estes possibilitam melhores resultados para técnicas de PLN na análise de informações não-estruturadas, já que os templates estruturam as informações em áreas pré-definidas, possibilitando uma busca mais eficiente pelos itens e links de rastreabilidade [29].

## 3.2. Informações não-estruturadas

Uma informação estruturada é uma informação na qual seu sentido não é ambíguo e é claramente representado em uma estrutura ou um formato de dado. O exemplo mais claro deste tipo de informação é uma tabela de um banco de dados relacional [03].

Já a informação não-estruturada é caracterizada como uma informação cujo sentido é só vagamente compreendido por sua forma e necessita de interpretação para que seja possivel extrair o seu sentido. Alguns exemplos são: documentos em linguagem natural, imagens e arquivos de áudio e vídeo [03]. Uma informação não-estruturada também pode ser considerada como uma informação sem contexto claramente definido, ao contrario das informações contidas em uma tabela de um banco de dados que pertencem à um contexto bem definido.

Na maioria das organizações de tecnologia da informação cerca de 80% das informações que são produzidas ou que ela contém estão gravadas de forma não-estruturada [03], em geral texto.

Em um processo de desenvolvimento de software também é possível encontrar uma porcentagem considerável de informações não-estruturadas. Por exemplo, no RUP, existem vários artefatos que são baseados neste tipo de informação, como: documento de visão, ECU, especificação de casos de teste, entre outros. A utilização de informações não-estruturadas é um dos maiores desafios para a automatização de um processo de rastreabilidade, já que elas contêm dados muito importantes para este processo.

Para análise e extração dessas informações são utilizadas várias técnicas combinadas. Dentre elas podemos citar: análise estatística, análise baseada em regras de processamento de linguagem natural, recuperação de informações, máquina de aprendizado e ontologias.

Uma das técnicas mais utilizadas para análise de documentos não-estruturados é o processamento baseado em conceitos [03]. Este tipo de processamento faz um uso extensivo de PLN. O PLN pode ser feito em vários níveis, a **Tabela 1** a seguir apresenta estes níveis para processamento de textos com informações não-estruturadas [03], bem como os seus respectivos objetos de análise e o resultado do processamento.

**Tabela 1. Níveis de análise de PLN em artefatos textuais.**

| Tipo | Objeto de Análise | Resultado |
|---|---|---|
| Morfológica | Palavras e variantes | Termos nos documentos |
| Pragmática | Sentidos de Contexto | Intenção das unidades textuais (Contexto de unidades) |
| Semântica | Relacionamento das palavras com o universo de conhecimento | Conceitos e relacionamentos |
| Estatística | Recorrência de termos | Nível de relacionamento entre os termos |
| Sintática | Ordem das palavras | Relacionamento entre os termos |

A abordagem proposta neste artigo utiliza a combinação de análise sintática, pragmática e semântica.

### 3.3. UML

Neste trabalho, a UML é utilizada como base para a extração das informações nos modelos de análise e projeto. Estes têm informações importantes para a criação dos links de rastreabilidade, além de serem a ligação entre os artefatos/itens criados na fase de requisitos com os da fase de implementação. Os modelos permitem também a extração de links que não ficam claros na fase de requisitos, o que aumenta a qualidade dos links gerados.

Um dos motivos mais fortes para a utilização de UML é a ampla adoção dela como linguagem de modelagem para sistemas de informação orientados a objetos. Outro fator importante é que a maioria das ferramentas de modelagem permite a exportação dos modelos para formato XMI, o que facilita o desenvolvimento de analisadores de maneira mais padronizada.

Por fim, a compatibilidade com outras ferramentas também foi um fator decisivo na adoção da UML como padrão na abordagem.

## 4. A Arquitetura proposta

A arquitetura de um sistema de gerência de informações não-estruturadas envolve três fases principais: aquisição, análise e a disponibilização de informações [03].

A fase de aquisição consiste no passo inicial de coletar um conjunto de documentos e transformá-los para um formato padrão, deixando-os disponíveis para a análise. Na fase de análise, a coleção de documentos adquiridos e os dados extraídos são analisados e os resultados são armazenados em estruturas bem definidas semanticamente. Já na fase de disponibilização, os resultados da fase de análise são disponibilizados juntamente com informações originais da fase de aquisição e outros dados estruturados, possibilitando assim o acesso ao usuário final através de aplicações apropriadas ou métodos definidos.

### 4.1. Aquisição

Na fase de aquisição, os artefatos gerados durante o processo são coletados e os seus dados são gravados em uma estrutura inicial, que tem como objetivo disponibilizar as principais informações para a fase de análise.

Os artefatos que seguem sintaxe precisa e rigorosa podem ser adquiridos de maneira simples devido a sua sintaxe. Já os artefatos que são predominantemente compostos por informações não-estruturadas, como os baseados em texto livre, devem seguir *templates* pré-definidos, algo comum na indústria, o que permite assim uma aquisição mais simples e com maior qualidade utilizando PLN.

Na aquisição, a ordem da coleta e extração dos dados é indiferente. Os artefatos são tratados de forma independente. O importante é extrair os dados e as relações explícitas nos documentos.

As regras de extração dos dados são parametrizadas, para que diferentes tipos de artefatos possam ser utilizados. Esta flexibilidade é importante, pois os *templates* dos artefatos evoluem naturalmente e diferem entre as organizações.

Dois tipos de informações são extraídas :

- Informações Básicas: informações para criação dos itens de rastreabilidade. Por exemplo: Necessidades, Funcionalidades, Classes, Casos de Teste, Unidades de Implementação e Artefatos;
- Informações Relacionais: informações que servem para a descoberta de relacionamentos, os links de rastreabilidade, e de seu sentido semântico. Por exemplo: Objetivos dos Casos de Uso, Especificações de Casos de Teste e Descrições de Classes.

Estas informações estão destacadas na Figura 3, que apresenta uma Especificação de Caso de Uso. As informações sublinhadas são informações básicas

existentes neste documento como o nome do Caso de Uso. Já as informações que estão em negrito informações relacionais. Nesta Figura, as informações básicas são: "GESTÃO DE ORDENS" e "REDIGIR ORDEM". Estas informações serão armazenadas como um item do tipo "Pacote" e "Caso de Uso" respectivamente. As informações relacionais serão armazenadas e relacionadas com o seu respectivo item e, posteriormente, serão utilizadas para estabelecer novos links ou para categorizar cada link com o seu sentido semântico. A Tabela 2 apresenta os artefatos e suas respectivas informações básicas e relacionais.

1. PACOTE: GESTÃO DE ORDENS

1.1 CASO DE USO: REDIGIR ORDEM

1.1.1. Breve Descrição

**Este caso de uso proporciona a elaboração de ordens de qualquer natureza para armazenamento ou envio às guarnições.**

**Figura 3. Especificação de Caso de Uso com informações identificadas.**

**Tabela 2. Informações Básicas e Relacionais de cada artefato de um processo de desenvolvimento de software.**

| Artefato | Informações Básicas | Informações Relacionais |
|---|---|---|
| Documento de Visão | Atores, Necessidades e Funcionalidades | Restrições |
| Especificação de Caso de Uso | Caso de Uso | Objetivos, Pré-condições, Fluxo Principal, Pós-condições e Outros Fluxos. |
| Especificação de Requisitos não-funcionais | Requisito Não-Funcional | |
| Modelo UML | Diagrama de Caso de Uso, Diagrama de Classes, Classes, Caso de Uso e Ator. | Relacionamentos e Descrições |
| Casos de Teste | Caso de Teste | |
| Código-fonte | Código-Fonte | |

Estas informações presentes na Tabela 2 formam o modelo intermediário que é a principal entrada para a fase de análise. A Figura 4 apresenta algumas informações extraídas e alguns links entre estas informações. As elipses representam informações básicas, e os retângulos, informações relacionais.

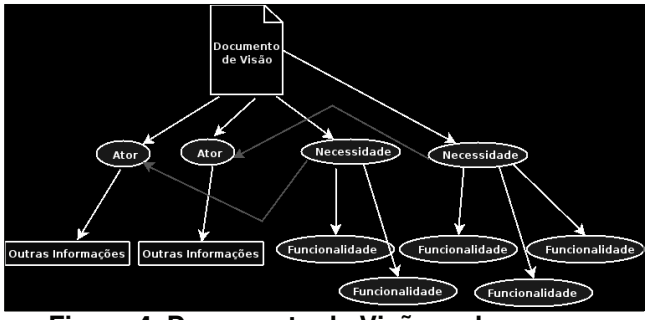


**Figura 4. Documento de Visão e algumas informações e relacionamentos extraídos do artefato.**

## 4.2. Análise

Na fase de análise os dados extraídos durante a fase de aquisição são utilizados com os objetivos de se organizar as informações de maneira estruturada, e descobrir e transformar estes dados em informações com sentido semântico definido. Esta fase contém a configuração das informações como ontologias, regras de processamento, dicionários, contextos de aplicação, idiomas, estruturas básicas, entre outras.

Diferentemente da fase de aquisição, onde a ordem da análise dos dados é indiferente, nesta fase ela será centrada nos modelos análise e projeto com o objetivo de ser compatível com o máximo de modelos de processos de software possíveis. Esta ordem não impede que a abordagem seja aplicada sem estes modelos desenvolvidos, porém para uma visão mais completa da rastreabilidade a construção dos mesmos é necessária.

Os dados disponibilizados pela fase de aquisição são utilizados de maneiras distintas na análise: as informações básicas são utilizadas para a criação dos itens de rastreabilidade; e as informações relacionais são utilizadas para criar links entre os itens criados, além de classificar esses relacionamentos.

Nesta abordagem, estes links de rastreabilidade utilizam-se das três dimensões [1]:

- Rastreabilidade vertical e horizontal: São as relações entre itens pertencentes ao mesmo modelo ou ao mesmo nível de abstração (vertical), ou em diferentes níveis de abstração (horizontal);
- Rastreabilidade explícita e implícita: Links explícitos são os pré-definidos. Links implícitos ocorrem onde não há relação explícita, porém eles podem ser deduzidos utilizando-se outras técnicas como o rastreamento por nomes em diferentes artefatos;

- Rastreabilidade estrutural e cognitiva: Links estruturais ocorrem quando há dependência sintática entre artefatos. Links cognitivos ocorrem pelo conhecimento semântico associado a decisões de projeto.

A rastreabilidade explícita é reconhecida, em sua maioria, na fase de aquisição. Os outros tipos de rastreabilidade são todos reconhecidos na fase de análise.

Para que estas visões sejam possíveis, na fase de análise as referências aos itens de rastreabilidade são reunidas e estruturadas, ao contrário do que acontece na aquisição, onde cada referência é armazenada de maneira independente. Esta tarefa organiza as informações e as estrutura para a fase de disponibilização.

### 4.3. Disponibilização

A fase de disponibilização é responsável por apresentar ao usuário final as visões geradas pelos tipos de rastreabilidades. Estas interfaces possibilitam, por exemplo, a geração de relatórios como o de análise de impacto. Elas devem também permitir que sejam iniciados novos processos de inferência, possibilitando assim a verificação, por exemplo, das dependências de um caso de uso ou de uma unidade de implementação.

As interfaces mais comuns nas abordagem são as matrizes de rastreabilidade. A Figura 5 apresenta um exemplo de matriz de rastreabilidade que descreve a relação entre Necessidades e Funcionalidades.

| | Funcionalidade 1 | Funcionalidade 2 | funcionalidade 3 |
|---|---|---|---|
| Necessidade 1 | | | |
| Necessidade 2 | | | |
| necessidade 3 | | | |

**Figura 5. Exemplo de matriz de rastreabilidade.**

Com o aumento dos tipos de links de rastreabilidade, a possibilidade de tipos de interface ou de mecanismos de visualização das informações de rastreabilidade aumenta bastante. Grafos, árvores, entre outros tipos de visualização, são indicados para apresentar essas novas visões e serão explorados em trabalhos futuros.

## 5. A Ferramenta

Para a implementação da abordagem descrita nos itens anteriores uma ferramenta foi construída obedecendo à arquitetura e aos critérios especificados.

De maneira geral ela funciona da seguinte forma: ela analisa todo o diretório do projeto realizando o *parser* dos arquivos presentes neste diretório e recolhendo os dados úteis para a criação dos rastros. Esta primeira etapa (fase de aquisição) é feita através de processamento de linguagem natural usando o framework UIMA (descrito na seção seguinte) e os dados provenientes dela são armazenados em registros sem relacionamentos em um banco de dados.

Na etapa seguinte (análise), os dados armazenados são analisados e os itens e links de rastreabilidade são criados. São gerados links tanto de rastreabilidade vertical, entre itens de um mesmo artefato, quanto horizontal, entre itens pertencentes a artefatos diferentes.

Rastreabilidade explícita e implícita também é abordada pela ferramenta. A primeira é simples de identificar, pois as informações já vêm definidas da fase anterior. A segunda é mais complicada e até o momento a implementação deste tipo de link consiste no rastreamento pelos nomes dos itens em diferentes artefatos.

Quanto à rastreabilidade estrutural e cognitiva, a ferramenta possui implementação apenas para a primeira.

Com a identificação destes links, as informações são salvas em um banco de dados estruturado para que possam refletir os rastros encontrados. É a partir deste banco de dados que são geradas as visualizações na fase seguinte (disponibilização). Estas visualizações até então consistem de matrizes de rastreabilidade. A seguir as fases brevemente descritas aqui serão melhor detalhadas.

### 5.1. Aquisição

Como descrito na seção da abordagem, esta fase consiste na coleta dos dados dos artefatos que serão usados na fase seguinte para a geração dos links de rastreabilidade.

Para implementar esta etapa optou-se pelo uso de *templates*, pois estes facilitam o processamento de linguagem natural em documentos que contém informações não-estruturadas. Os *templates* utilizados pertencem ao RUP, mas a ferramenta permite que sejam escritas regras para outros modelos de processos.

Para a tarefa de extração de informações dos artefatos, a ferramenta utiliza o framework UIMA (Unstructured Information Management Architecture) [03], cuja principal característica é o desenvolvimento rápido e padronizado de arquiteturas flexíveis, extensíveis e reutilizáveis para gerenciamento de informações não-estruturadas.

O UIMA é uma plataforma comum para aplicações de gerenciamento de informações não-estruturadas focando, principalmente, na aquisição e análise destas informações. Seu objetivo final é transformar informações não-estruturadas em informações estruturadas por meio de um controle e sincronismo de

componentes de análise de informações não-estruturadas, para detecção dos itens que constituirão as informações estruturadas e, desta maneira, construir uma ligação entre as informações não-estruturadas e estruturadas [03, 05]. Assim, o UIMA consiste de sistemas de software que analisam informações não estruturadas para extrair, organizar e entregar dados que sejam relevantes para o cliente ou para a aplicação final [04].

Neste contexto, a estratégia adotada para extração das informações é a utilização do UIMA para desenvolver uma arquitetura que tenha as seguintes características:

1. Extração de informações em camadas, levando em consideração o contexto da informação;
2. Orientada a componentes, sendo flexível para receber novos analisadores de informação; e
3. Flexível em relação às regras de extração dos itens, podendo ser configurada para mais de um tipo de *template* e, se possível, em alto-nível.

A análise proposta pelo UIMA é baseada na criação de blocos básicos chamados de *Analysis Engines* (AEs) que analisam um documento e inferem e gravam atributos descritivos sobre ele na forma de metadados. AEs são construídos a partir de blocos menores chamados de *Annotators*. Um *annotator* é um componente que contém a análise lógica. É ele que irá guardar todo o algoritmo de análise, ou seja, que conterá o código que fará o *parsing* dos dados para assim extrair as informações relevantes criando dados adicionais (metadados) sobre o artefato que está sendo analisado [06].

Cada item de rastreabilidade tem um ou mais AEs que analisa os dados de entrada para extração das informações importantes para construção dos links de rastreabilidade. As definições básicas de um AE são feitas utilizando XML (*eXtensible Markup Language*). Esta configuração permite mais de uma regra de análise, possibilitando assim analisar diferentes modelos de documentos em um mesmo AE.

Os dados de entrada dos AEs da ferramenta são os artefatos gerados no desenvolvimento de um sistema, como documentos de especificação, modelos UML e código fonte. Até o presente momento, a ferramenta já realiza esse *parsing* das informações em documentos de especificações e em modelos UML.

O framework UIMA fornece uma implementação em java para o desenvolvimento dos reconhecedores. Os componentes foram desenvolvidos nesta linguagem e configurados em arquivos XML, o que possibilita a modificação de regras sem a necessidade de uma nova compilação do *Annotator*.

Para os documentos de especificações, o parser foi feito diretamente, pois estes estão em forma de texto. Já os artefatos UML foram primeiramente exportados para o padrão XMI e só então o parser foi realizado nestes arquivos para que os elementos importantes para a rastreabilidade fossem identificados.

As classes dos *annotators* chamam os métodos que iniciarão a persistência dessas informações. Para realizar essa persistência usou-se JPA (*Java Persistence API*), que é a especificação para gerenciamento de persistência e mapeamento objeto-relacional [07]. JPA provê facilidades para o mapeamento objeto-relacional por gerenciar dados relacionais em aplicações Java [12]. Assim, as informações capturadas são armazenadas em um banco de dados simples e posteriormente elas são reorganizadas para criar os links de rastreabilidade do sistema.

## 5.2. Análise

Na fase de aquisição os dados importantes são encontrados através do *parser* e armazenados em um banco de dados em entidades não relacionadas. A partir destas informações, o processo de análise é iniciado. Estas informações são adequadas ao modelo estruturado, criando assim os itens e posteriormente os links de rastreabilidade. O modelo do banco de dados estruturado é apresentado na Figura 6.

Os tipos dos itens (tabela typeItem) são os tipos listados nas informações básicas na Tabela 2. Assim, quando um caso de uso é encontrado na análise, um Item é criado, suas informações são persistidas e o seu tipo é "Caso de Uso".

As informações relacionais, bem como outras incidências do mesmo item em outro artefato, são utilizadas para criação dos links e a definição de seu sentido (tabela typeLink).

Por exemplo, se um Caso de Uso é encontrado na Especificação de Caso de Uso e posteriormente no Diagrama de Caso de Uso, um link é criado e seu tipo é de "Evolução".

No modelo de dados existem ainda outras entidades como useCase, actor e classItem que são especializações de um Item.

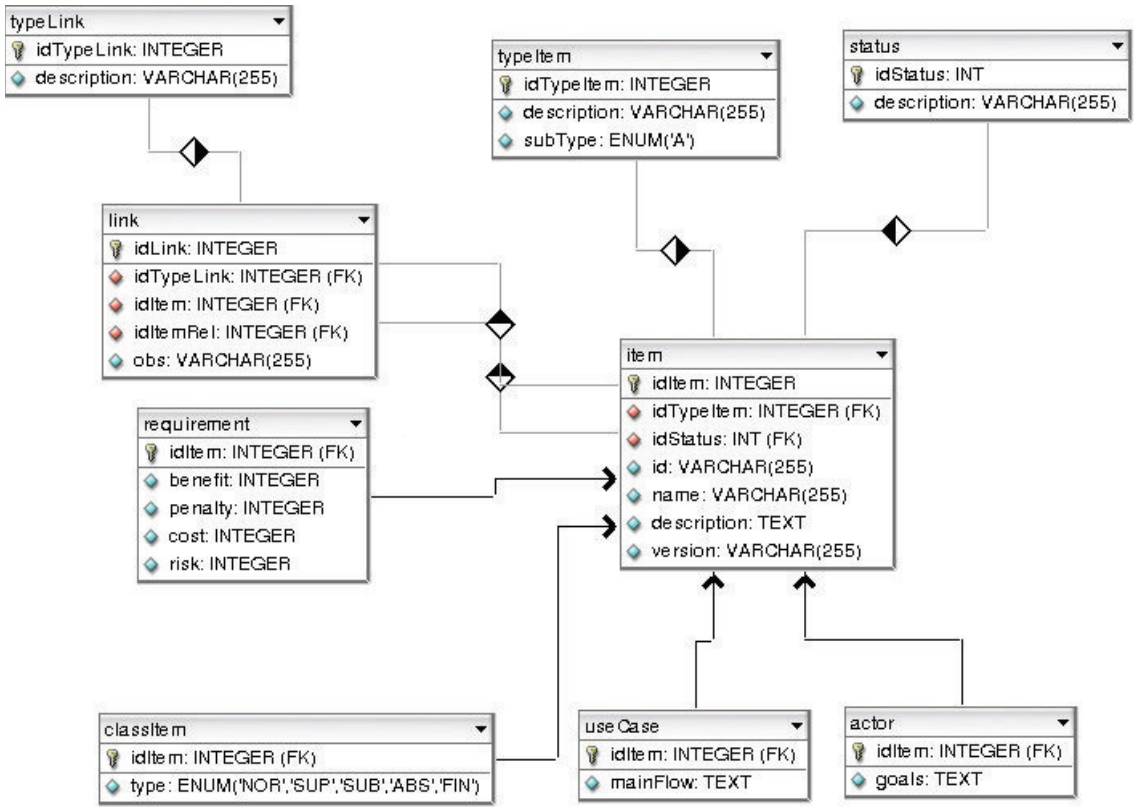


**Figura 6. Modelo do banco de dados estruturado.**

### 5.3. Disponibilização

A etapa de disponibilização consiste na apresentação dos resultados para o usuário através de visualizações. Para visualização de um resultado inicial ou intermediário, o UIMA disponibiliza um componente chamado *Document Analyser*, no qual é possível testar se as regras de extração dos dados estão encontrando as informações corretas. A Figura 7 apresenta o resultado de um AE em um documento de visão. À esquerda há o conteúdo do documento com as marcações feitas pelos *annotators*, e à direita são apresentadas as informações extraídas pelos *annotators*. Estas marcações feitas pelo UIMA se diferenciam pela cor: cada *annotator* utiliza uma cor para marcação que são visualizadas no componente *Document Analyser*.

A partir dos dados armazenados em uma forma estruturada na etapa de análise é possível criar visualizações que mostrem a rastreabilidade do sistema e permitam uma análise de impacto. Neste primeiro momento matrizes de rastreabilidade simples foram utilizadas para esta visualização. Um exemplo de uma destas matrizes encontra-se na Figura 5.

A ferramenta gera hoje as seguintes matrizes de rastreabilidade:

- Necessidade X Funcionalidade
- Funcionalidade X Ator
- Ator X Caso de Uso
- Caso de Uso X Caso de Uso
- Caso de Uso X Caso de Teste
- Caso de Uso X Diagrama de Classes
- Classe X Classe

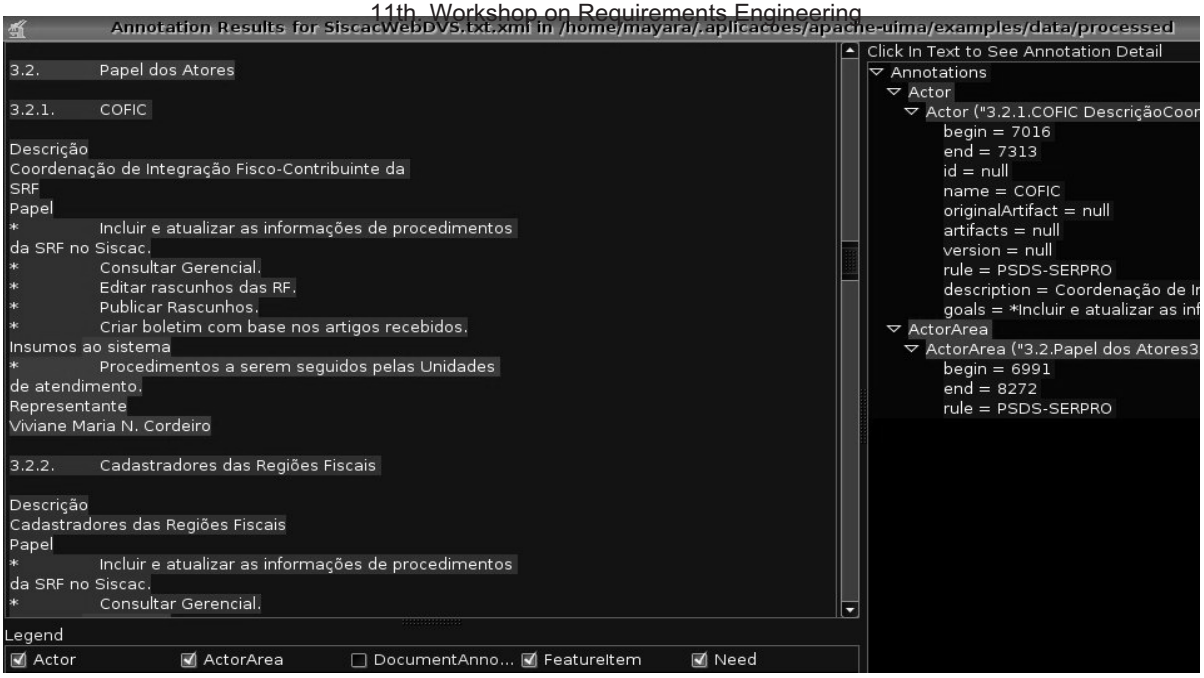


**Figura 7. Resultado de um AE apresentado pelo Document Analyser.**

## 6. Considerações finais e trabalhos futuros

A Ratreabilidade de Software (RS) é uma prática muito útil e importante no desenvolvimento de sistemas de software. Ela permite ter um maior controle na evolução dos requisitos, realizar uma análise de impacto e de riscos de mudanças muito relevante para o planejamento dos passos do projeto, além de trazer melhoras à documentação do sistema, facilitando assim sua manutenção. Portanto, a RS é uma prática muito desejável nos projetos, ainda que pouco utilizada.

A abordagem apresentada neste artigo tem como principal característica a formação e manutenção dos links de rastreabilidade de forma semi-automática para diminuir o esforço e o custo de realização dessa prática e assim possibilitar que ela seja mais utilizada em projetos de software. Trata-se de uma abordagem genérica que pode ser implementada satisfazendo as características de cada projeto.

Uma ferramenta foi implementada para demonstrar uma maneira de aplicar a abordagem. Este protótipo ainda será refinado e mais extensivamente testado, mas já demonstra a viabilidade de se aplicar rastreabilidade de software semi-automática no desenvolvimento de sistemas.

Como trabalhos futuros podem se destacar: a comparação de ferramentas; utilização em mais de um tipo de modelo de processo; a automatização de mais tipos de artefatos do processo; e a utilização de mais tipos de links de rastreabilidade e a geração de novas formas de visualização dos resultados. Além disso, pretende-se avaliar a nossa abordagem em um ou mais estudos de caso reais visando avaliar a sua precisão e a sua acurácia em contraste com outras abordagens.

### Agradecimentos

Esta pesquisa tem financiamento da Universidade Federal do Pará, do CNPq através do Edital Universal 2006/2, processo 479206/2006-6 e da IBM através de um IBM UIMA Innovation Award.

## 7. Referências

[01] Bianchi, A., Visaggio, G., and Fasolino, A. R. (2000). An exploratory case study of the maintenance effectiveness of traceability models. In IWPC '00: Proceedings of the 8th International Workshop on Program Comprehension, page 149, Washington, DC, USA. IEEE Computer Society

[02] SUÍTE RATIONAL. Rational Software Corporation. [s.l.] Disponível em: <http://www-306.ibm.com/software/rational/>. Acesso: Nov. 2007.

[03] Ferrucci, D.; Lally, A. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. Natural Language Engineering , 2004.

[04] Apache UIMA Development Community. UIMA Overview & SDK Setup. Dezembro 2007.

[05] UIMA: Unstructured Information Management Architecture. AlphaWorks IBM, [s.l.]. Disponível em:

<http://www.alphaworks.ibm.com/tech/uima>. Acesso em: Ago. 2007.

[06] Apache UIMA Development Community. UIMA Tutorial and Developers' Guides. Dezembro 2007.

[07] Creative Commons Attribution. Java Persistence API. Disponível em: http://jpa.wikidot.com. Acesso em: 11 Abr 2008.

[08] Spanoudakis, G.; Zisman, A. Software Traceability: A Roadmap. Advances in Software Engineering and Knowdledge Engineering, Vol. 3: Recent Advances,(ed) S.K Chang, World Scientific Publishing, ISBN:981-256-273-7, Ago. 2005.

[09] Lindval M.; Sandahl K. Practical Implications of Traceability. Software Practice and Experience, vol. 26, no. 10, p. 1161-1180, 1996.

[10] Ramesh B.; Jarke M. Towards Reference Models for Requirements Traceability. IEEE Transactions in Software Engineering, 27(1), 58-93, 2001.

[11] Arkley P.; Mason P.; Riddle S.; Position Paper: Enabling Traceability. Proceedings of 1st International Workshop on Traceability in Emerging Forms of Software Engineering, 61-65, Disponível em: <http://www.soi.city.ac.uk/~zisman/traceworkshop.html>. Acesso em Dez. 2006.

[12] SunMicrosystems, Inc. Java EE Tutorial. Setembro 2007.

[13] Pohl, K. Process-Centered Requirements Engineering. John Wiley Research Science Press, 1996.

[14] Antoniol, G.; et all. Recovering Traceability Links between Code and Documentation. IEEE Transactions on Software Engineering, 28(10), 970-983, Out. 2002.

[15] Marcus A.; Maletic J.I. Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing. International Conference Software Engineering, v. 25th, 2003.

[16] Spanoudakis, G., et all. Rule-Based Generation of Requirements Traceability Relations. Journal of Systems and Software, 72(2), p. 105-127, 2004.

[17] Zisman A., et all. Tracing Software Requirements Artefacts. Proceedings of the 2003 International Conference on Software Engineering Research and Practice, Las Vegas, Nevada, USA, Jun. 2003.

[18] Cleland-Huang J.; Schmelzer D. Dynamic Tracing Non-Functional Requirements through Design patter Invariants. Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2003), Canada, Out. 2003.

[19] Egyed A.; Gruenbacher P. Automatic Requirements Traceability: Beyond the Record and Replay paradigm. Proceedings of the 17th IEEE International Conference on Automated Software Engineering (ASE), Edinburgh, UK, Set. 2002.

[20] Kelleher, J.; Simonsson, M. Utilizing use case classes for requirement and traceability modeling. Proceedings of the 17th IASTED international conference on Modelling and simulation, p. 617–625, Anaheim, USA. ACTA Press, 2006.

[21] Cleland-Huang, J. Requirements traceability - when and how does it deliver more than it costs? Proceedings of the 14th IEEE International Requirements Engineering Conference, p. 323, Washington, USA. IEEE Computer Society, 2006.

[22] Salem, A. Improving software quality through requirements traceability models. Computer Systems and Applications - IEEE International Conference, p. 1159–1162, Washington, USA. IEEE Computer Society, 2006.

[23] Letelier, P. A framework for requirements traceability in uml-based projects. 17th IEEE International Conference on Automated Software Engineering.

[24] Kotonya & Sommerville. Requirements Engineering: Processes and Techniques. John Willey & Sons Ltd, 1998.

[25] Spence I., Probasco L. Traceability Strategies for Managing Requirements with Use Cases. Rational Software White Paper, 2000.

[26] Emam, E. Causal Analyses of the Requirements Change Process for a Large System. IESE-Report No. 054.97/E, 1997.

[27] Jones. Assessment and Control of Software Risks. Prentice Hall, 1994.

[28] Blackburn, M.; Busser, R.; Nauman, A. Removing Requirement Defects and Automating Test. Software Productivity Consortium NFP. Disponível em <http://www.software.org/pub/taf/downloads/RemovingRequirementDefects.pdf>. Acesso em: Nov. 2002.

[29] Kelleher, J. Tracebility Patterns. Proccedings of ECMDA Tracebility Workshop, Bilbao, 2006.

[30] Lee, R. C.; Tepfenhart, W. M. UML e C++ - Guia de desenvolvimento orientado a objeto, São Paulo: Ed. Makron Books, 2002.

[31] Davis, A. M. Software Requirements: Objects, Functions and States. Englewood Cliffs, New Jersey: Prentice Hall. 1993.

[32] Pohl K. PRO-ART: Enabling Requirements Pre-Traceability. Proceedings of the 2nd IEEE International. Conference on Requirements Engineering, 1996.

[33] Cleland-Huang J.; Chang C.; Wise J. Supporting Event Based Traceability through High-Level Recognition of Change Events. Proceedings of IEEE COMPSAC Conference, Oxford, England, August 2002.

[34] Gotel O.; Finkeltein A. Contribution Structures. Proceedings of 2nd International Symposium on Requirements Engineering, pag. 100-107, 1995.