Rastreabilidade Semi-Automática Através do Mapeamento de Entidades

Jerônimo Backes[1], **Daltro José Nunes**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

Resumo. *Entre os fatores que geram o alto custo da rastreabilidade, está a dificuldade na criação e manutenção de relações precisas entre artefatos. Praticamente todas as metodologias existentes preocupam-se em relacionar artefatos diretamente entre si, o que dificulta o uso de processos automatizados na derivação de novos relacionamentos, bem como na manutenção dos já existentes. Com base nestas observações, o presente trabalho propõe o uso de estruturas intermediárias nos relacionamentos, chamadas de entidades, para representar os interesses tratados pelos artefatos, e derivar, automaticamente, relações complexas entre os mesmos. Este modelo foi avaliado por profissionais da indústria e apresentou-se como solução de rastreamento viável, em comparação com as tradicionais matrizes de rastreabilidade. Espera-se que este sirva como base para soluções inovadoras na área, que ainda é considerada problemática.*

Abstract. *The difficulties in creation and maintenance of precise relationships between artifacts are the root cause of traceability high cost. Virtually all existing methodologies propose solutions that relate artifacts directly between themselves, what hinders the use of automated processes to derivate new relationships, as well as maintaining existing ones automatically. Based on these observations, this paper proposes the use of intermediate structures, called entities, to represent the interests present on traced artifacts. These entities can be used to derive, automatically, complex relationships between artifacts. The proposed model was evaluated by industry professionals and was considered by them as a viable solution for traceability when compared to the traditional traceability matrices. It is believed that this model will serve as a basis for innovative research solutions in traceability, which is still considered a problematic field.*

1. Introdução

No desenvolvimento de software, uma vasta gama de artefatos é gerada e mantida: documentação, requisitos, modelos de projeto, casos de teste, etc. Todos utilizados para compreender melhor o sistema [Kelleher and Simonsson 2006]. O objetivo do gerenciamento de requisitos e da rastreabilidade é, durante o ciclo de vida do produto, organizar, analisar, e rastrear os requisitos que estes artefatos satisfazem [Delgado 2006]. O rastreamento pode ser visto como um investimento na tentativa de tornar explícitos e usáveis os relacionamentos entre requisitos e demais artefatos, servindo como base para atividades de gerenciamento e desenvolvimento [Heindl and Biffl 2006].

Não é necessário discutir o valor potencial que esta atividade é capaz trazer ao desenvolvimento de sistemas, afinal, a literatura relata que um dos maiores problemas na tarefa de manutenção está relacionado à compreensão de como o sistema funciona: alguns estudos apontam que de 30% a 60% dos custos nesta fase estão relacionados à descoberta do que o sistema faz [Tryggeseth and Nytro 1997]. Contudo, a rastreabilidade é raramente utilizada, já que em muitos casos o custo envolvido não cobre os benefícios. Este alto custo vem: (1) da dificuldade em gerar automaticamente as relações, com uma semântica clara e precisa; (2) da heterogeneidade e do grande número de artefatos que são criados durante o desenvolvimento; e (3) a falta de corretude e completude das relações de rastreabilidade [Cleland-Huang 2006].

Foram analisadas diversas propostas, em cerca de 250 artigos, tratando do estabelecimento e manutenção de relacionamentos, e percebeu-se que praticamente todas preocupam-se com a criação de relações diretas entre requisitos e artefatos. Porém, constatou-se que relacionamentos diretos ocultam os interesses que ambos artefatos tratam, dificultando a identificação das razões que motivam a existência dos elos, bem como inibindo a eficácia de processos automatizados para criação e manutenção dos mesmos. As decisões sobre o estabelecimento ou não de relacionamentos entre artefatos devem ser tomadas pelo usuário [Marcus et al. 2005], porém, assegurar uma qualidade aceitável nos elos, sem pressionar demasiadamente os desenvolvedores, é um problema desafiador, que carece de maiores pesquisas [Neumuller and Grunbacher 2006]. A grande maioria das propostas na área utilizam uma das duas técnicas:

1. **Rastreamento Manual:** geralmente usando matrizes de rastreabilidade, que simplesmente exibem relacionamentos entre artefatos. As tentativas de sua utilização pela indústria, resultaram, quase que invariavelmente, em fracaso [Cleland-Huang 2006]. As informações mantidas geralmente restringem-se a simplesmente "um artefato está relacionado a outro", onde qualquer relacionamento pode ser criado, causando grande dificuldade em identificar elos incorretos e até mesmo identificar as razões de sua existência. Outro grave problema é a dificuldade e custo de realizar o rastreamento com alto nível de detalhe.
2. **Recuperação Automática de Relacionamentos:** geralmente utilizando técnicas de indexação LSI (Latent Semantic Indexing), que simplesmente relacionam porções texto, de especificações de requisitos e manuais, com identificadores do código fonte – uma tarefa típica e reconhecimento de padrões [Penta et al. 2002] – produzindo relacionamentos com as mesmas características citadas no item anterior, com o agravante de que a maioria dos elos gerados devem ser descartados pelo usuário, por não possuírem sentido [Lucia et al. 2006]. Não obstante, erros de decisão são muito comuns, pois o analista acaba descartando elos verdadeiros, e mantendo elos incorretos [Hayes and Dekhtyar 2005]. A dificuldade na identificação da natureza das relações pode ser um dos diversos fatores que ocasionam este fenômeno.

Com isto, o resultado é geralmente o mesmo: dificuldade na manutenção dos relacionamentos e sua conseqüente degradação, que por fim se traduz em tempo, esforços e recursos completamente desperdiçados. Assim, o presente trabalho apresenta uma forma de estabelecimento de relações com base em estruturas intermediá-

rias, que representam interesses em comum representados por diferentes artefatos. Estas estruturas, chamadas de entidades, têm por objetivo centralizar interesses para derivar relacionamentos entre os artefatos. Esta estratégia permite a criação e manutenção da estrutura de rastreabilidade de forma semi-automática, tornando-a mais precisa e de produção menos custosa.

Este artigo está organizado da seguinte forma: a seção 2 apresenta as técnicas mais utilizadas na área e seus maiores problemas; já seção 3 trata dos problemas relacionados à manutenção da estrutura de rastreabilidade. A proposta de rastreabilidade é apresentada na seção 4, e sua avaliação na seção 5. A seção 6 apresenta trabalhos relacionados e os compara com a proposta deste trabalho. A seguir, são apresentadas várias sugestões de trabalhos futuros, na seção 7. Finalmente a seção 8 apresenta a conclusão do presente artigo.

2. Contextualização

Tradicionalmente, as relações de rastreabilidade são criadas conforme a relação matricial (RM): $\{A_n\ relatedTo\ A_m\ | A$ é um artefato e n e m são os modelos utilizados para defini-los (eg. caso de uso, especificação de requisitos, diagrama de classe, código fonte de uma linguagem específica, etc $\}$. Neste trabalho o termo "artefato" corresponde estritamente a todo modelo que representa a arquitetura sistema em algum nível de abstração (diagramas de classe, de seqüência, modelos entidade-relacionamento, etc), casos de teste, ou código fonte, sendo este último o modelo concreto do sistema [Maletic et al. 2005]. A função $relatedTo$ relaciona um artefato a outro, criando "elos" de rastreabilidade.

Cada artefato possui um nível de abstração, que pode ser mais ou menos completo do que os outros, e apresenta diferentes pontos de vista sobre as necessidades do sistema. As relações entre estes, do menos completo para o mais completo, até a implementação, deve ser mantida para garantir que a estrutura de rastreabilidade permita: (1) o mapeamento dos requisitos para um modelo posterior, até o código fonte, garantindo que a satisfação dos requisitos esteja atribuída à componentes do sistema (*Forward from Requirements*); (2) o Mapeamento de um modelo qualquer de volta para os requisitos, evitando o *gold-plating*(*Backward to Requirements*);

Grande parte das propostas de rastreabilidade possuem estas características, no entanto, algumas suportam o rastreamento somente vertical (relacionamentos entre artefatos no mesmo nível de abstração), outras somente o horizontal (relacionamentos entre artefatos em níveis diferentes de abstração). Para exemplificar a utilização de matrizes no rastreamento vertical, considere os seguintes requisitos:

R1 "Todo cliente tem uma conta, com um determinado limite de crédito";
R2 "O limite de crédito de qualquer conta está restrito a um determinado valor, estabelecido segundo a lei federal XYZ";
R3 "A verificação do limite de crédito é realizada de acordo com os seguintes critérios...";
R4 "Universitários têm limite de R$ 200,00, invariavelmente".

Não é necessário muito esforço de interpretação para perceber que os requisitos estão de fato relacionados, pois tratam de interesses comuns, formando a possível

Tabela 1. Matriz de rastreabilidade entre os requisitos R1 a R4

Requisitos	R1	R2	R3	R4
R1	■	X		X
R2	■	■	X	
R3	■	■	■	X
R4	■	■	■	■

matriz de rastreabilidade, representada pela Tabela 1.

Note que apesar do pequeno número de requisitos e relacionamentos, não é trivial interpretar a matriz da Tabela 1 e identificar o motivo por trás da existência dos elos e a inexisência de outros. No caso do exemplo, estes foram criados pelas seguintes interpretações:

R1-R2 "A lei federal XYZ estabelece o limite da conta, citado em R1";
R1-R4 "O limite da conta também pode ser definido pelo valor citado em R4";
R2-R3 "A verificação deve considerar as regras impostas pela legislação";
R3-R4 "A verificação deve considerar o caso das contas universitárias".

Porém, várias interpretações diferentes podem ser tomadas a partir dos requisitos apresentados, formando relacionamentos diferentes. Assim, para evitar este esforço de re-interpretação, é necessário anexar a cada elo as decisões, problemas, suposições e argumentos que motivaram sua criação. Analogamente, é fundamental que as mesmas informações sejam mantidas para os elos que não foram criados, evitando o esforço desnecessário de justificar novamente uma decisão já estabelecida [Ramesh and Jarke 2001].

Para exemplificar o uso de matrizes no rastreamento horizontal, considere os requisitos R1 a R4, "implementados" no código fonte da listagem 1. Como a listagem implementa os requisitos, pode-se dizer que estes pertencem ao conjuntto de relações $\{ A_{requisitos}\ relatedTo\ A_{codigo} \}$.

Listagem 1. Exemplo de implementação dos requisitos

```
1   Client{
2           personalData;
3           account;
4   }
5
6   Account{
7           cash;
8           limit;
9           type;
10  }
11
12  System{
13          listOfClients;
14
15          limitVerification(Amount a, Account c){
16                  ...
17          };
18
19          withdraw(Client c, Account acc, Amount a){
20                  limitVerification(a, acc);
21                  ...
22          };
23  }
```

A Tabela 2 representa as possíveis relações entre os requisitos e partes do código apresentado. O requisito R1 é satisfeito pela implementação das classes *Client* e *Account*, bem como pelo atributo *limit* em *Account*. R2 trata do limite da conta, então a implementação de *Account.limit* pode ser afetada caso R2 seja alterado (e vice-versa). Já R3 é satisfeito pela implementação de *System.limitVerification*, e R4 pelo tratamento de um tipo de conta (Account.type, que pode ser "conta de pessoa física", "conta de pessoa jurídica", "conta universitária", etc). Note que a matriz de rastreabilidade depende muito da interpretação humana, sendo que outras interpretações e necessidades podem originar uma matriz completamente diferente.

Tabela 2. Matriz de rastreabilidade entre requisitos e código.

Requisito	Classes	Métodos	Atributos
R1	Client, Account		Client.account.limit
R2	Account		Account.limit
R3		System.limitVerification	
R4	Account		Account.{limit, type}

Analisando mais profundamente, não é difícil identificar o potencial de crescimento de uma matriz de rastreabilidade. Os problemas de manutenção aparecem quando tenta-se atualizar os elos, especialmente quando se rastreia um grande número de artefatos, com um nível de granularidade que possa ser considerado útil, como no exemplo da Tabela 2. Por exemplo, projetos com centenas de casos de uso, compostos por inúmeros passos, e relacionados a outros artefatos, são de manutenção extremamente complexa e cara. A matriz de rastreabilidade facilmente assume um tamanho ingerenciável, pois a quantidade de relações tende a crescer exponencialmente [Cleland-Huang et al. 2004]. A partir deste ponto é que as técnicas de rastreabilidade baseadas em matrizes falham. Um dos maiores problemas reside na incapacidade de lidar eficientemente com alterações que afetam os relacionamentos estabelecidos [Maletic et al. 2005].

3. Problemas da Rastreabilidade

Os métodos de rastreabilidade existentes focam-se na criação de estruturas de relacionamentos. Embora úteis, o problema é evoluir estas estruturas [Murta et al. 2006]. Entre as soluções mais aplicadas para encontrar o melhor *tradeoff* entre o tamanho e a utilidade das informações de rastreabilidade, estão rastrear somente requisitos críticos e reduzir a granularidade (por exemplo, a tabela 2 poderia relacionar somente requisitos e classes). Infelizmente, nenhuma destas é satisfatória, já que uma pequena alteração pode afetar diversas partes de um sistema, e a análise de impacto deve contar com o maior número de detalhes possível para que os efeitos colaterais possam ser detectados com precisão. Além disso, muitos esquemas de rastreabilidade fornecem pouco suporte para identificar o impacto de requisitos totalmente novos [Cleland-Huang et al. 2002].

A maioria das ferramentas e métodos de rastreabilidade suportam a identificação de artefatos afetados, quando existe um conjunto preciso de elos. Contudo, grande parte destes não fornece nenhum tipo de suporte para assegurar que os artefatos e elos relacionados, afetados por alguma alteração, sejam atualizados rapidamente. A atualização é tipicamente manual, e como resultado, surgem erros. Estes podem afetar a capacidade de, futuramente, identificar os impactos de novas alterações [Cleland-Huang et al. 2003].

Tais limitações praticamente impossibilitam o uso da rastreabilidade fácil e eficientemente, portanto, o presente trabalho propõe um novo modelo de rastreamento, cujo objetivo é mitigar os problemas relacionados à dificuldade de criação e manutenção de estruturas de rastreabilidade com maior precisão e utilidade.

4. Modelo proposto

Analisando a relação matricial, apresentada no início da seção 2, constatou-se que o maior problema das técnicas de rastreabilidade está na dificuldade em indentifi-

car os motivos pelos quais as relações existem. A simples informação de que um artefato a está relacionado a outro exige grande esforço de interpretação humana para identificar esses motivos. Por conta disso, as técnicas de recuperação automática também não são eficientes, afinal, não é possível esperar que um computador interprete corretamente o significado de textos produzidos em linguagem natural e crie elos consistentes. A utilização de técnicas automatizadas também oculta dos motivos por trás da existência das relações.

Para mitigar estes problemas, e considerando especialmente as dificuldades no estabelecimento e manutenção de uma estrutura de rastreabilidade, decidiu-se introduzir um modelo cujo objetivo é facilitar o processo criação de elos, semi-automatizando parte do processo. Espera-se que este reduza o esforço de interpretação necessário para identificar as relações, ao mesmo tempo que permita atualizações com o mínimo possível de intervenção humana.

4.1. Visão geral

O processo de desenvolvimento de software compreende a criação de diferentes abstrações da realidade, usadas para representar e controlar a arquitetura do sistema, avaliar oportunidades e identificar riscos. O termo "modelo" é geralmente usado para se referir à estas abstrações [Noll and Ribeiro 2007]. Um modelo de requisitos, por exemplo, expressa idéias, conceitos. Estes conceitos, relacionados em um contexto, formam um significado. Para facilitar o entendimento deste significado, pode-se usar vários modelos, em um ou mais níveis de abstração, para expressar diferentes pontos de vista.

O modelo proposto visa o rastreamento desses modelos, e conseqüentemente, de diferentes níveis de abstração e pontos de vista sobre um determinado interesse. Como resultado, torna-se possível identificar onde o interesse é satisfeito, qual a influência de outros interesses sobre este, etc. Por exemplo, o requisito R1, na seção 2, trata dos conceitos de "cliente", "conta", "crédito" e "limite". Estes são relacionados de forma a sugerir a idéia de que "O cliente tem uma determinada conta, com um determinado crédito, que está restrito a um determinado limite". Basicamente, um requisito como: "o cliente não pode creditar um valor superior ao limite de sua conta", relaciona os conceitos supracitados, de forma a expressar uma necessidade que o sistema deve atender.

Os demais artefatos que compõe o sistema devem satisfazer os requisitos, e portanto, certamente tratam dos mesmos interesses de alguma forma. Assim, ao mapear os interesses identificados nesses artefatos, deverá ser possível derivar, automaticamente, relações complexas entre artefatos, utilizando como referência os interesses que compartilham. Dessa forma, pode-se relacionar todos os artefatos e seus elementos aos interesses que representam, permitindo recuperar elos de rastreabilidade implícitos [Noll and Ribeiro 2007].

O presente trabalho propõe um modelo de rastreabilidade baseado na identificação destes interesses, que doravante serão denominados entidades. O modelo permite registrar as áreas de interesse que dão origem às entidades, bem como classificá-las de acordo com suas características. A Figura 1 apresenta a proposta na forma de um modelo de entidade-relacionamento (ER).

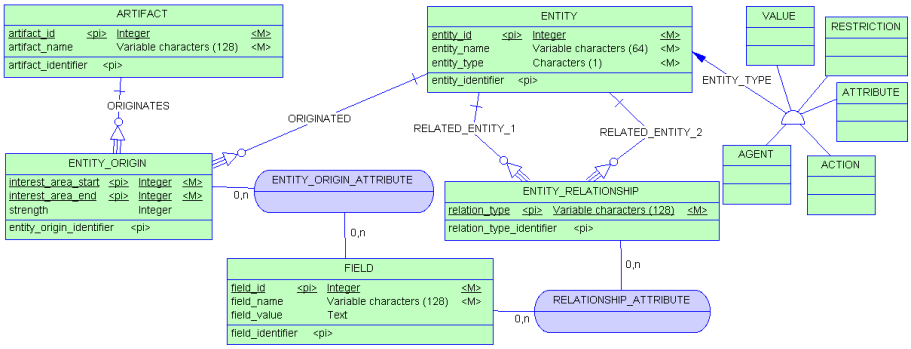


Figura 1. Modelo de rastreabilidade proposto

Conforme a Figura 1, os artefatos possuem áreas de interesse que identificam entidades. Estas podem ser classificadas dependendo da influência que exercem umas sobre as outras. A seção 4.2, a seguir, discorre sobre estas mais detalhadamente. Note que uma entidade pode ser relacionada à diversos artefatos. Como será visto na seção 4.3, cada relação identifica uma área de interesse em algum artefato, que dá origem à alguma entidade. Além disso, entidades podem estar relacionadas entre si, e pode-se definir tipos de relacionamentos para dar maior significado a estes relacionamentos. Relacionamentos podem possuir atributos e valores, estes definidos conforme a necessidade: pode-se registrar as motivações que justificaram a criação do relacionamento, marcá-lo como inválido ou exigindo verificação, o responsável pela criação da relação, datas, etc. Finalmente, conforme apresentado na seção 4.3.1, o modelo proposto prevê alguns tipos básicos de relacionamento, sem impedir que outros sejam definidos.

4.2. Entidades

O modelo proposto foi inspirado no trabalho de Kelleher e Simonsson [Kelleher and Simonsson 2006], que utiliza classes UML para modelar e rastrear requisitos. No entanto, deseja-se não impor restrições sobre a forma pela qual os requisitos são modelados, nem afetar o processo de desenvolvimento, conforme sugerem Neumuller e Grundbacher [Neumuller and Grunbacher 2006]. Além disso, o modelo deve ser utilizável sobre quaisquer artefatos. Assim, decidiu-se formular um modelo mais abstrato, que não influencie a modelagem do sistema, e que permita identificar interesses relacionados em diversos artefatos. O modelo de entidades estende o proposto por Jiang et al [yi Jiang et al. 2007], que usa centralizadores de elos de rastreabilidade (denominados âncoras), identificando áreas de interesse comuns a mais de um artefato, conforme a Figura 2. Entidades, além de atuarem como centralizadores, são usadas para abstrair, classificar e relacionar as áreas de interesse identificadas nos requisitos.

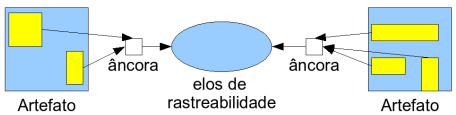


Figura 2. Âncoras representam áreas de interesse dos artefatos

As entidades podem ser classificadas em:

Tabela 3. Relacionamentos entre as entidades identificadas

Agente	Cliente	Conta	Lei XYZ
Atributos	conta	limite	
Ações		verificação limite	
Restrições		restrição limite	
Ação	verificação limite		
Baseada em	critérios		
Afeta	limite		
Restrição	restrição limite		
Baseada em	lei XYZ, cliente, conta, limite		
Restringe	limite		

- **Agentes** causadores de modificações sobre o estado do sistema (eg. cliente);
- *Atributos* pertencentes a agentes, que contêm algum valor (eg. saldo, limite);
- ***Valores*** de atributos (eg. R$200, status bloqueado, etc);
- Ações que manipulam os atributos de um ou mais agentes (eg. creditar, debitar, verificar limite);
- Restrições que identificam limites para os valores dos atributos de um ou mais agentes, ou restringem suas ações (eg. saldo não pode ser inferior ao limite);
- `Entidades sem classificação definida` usadas para simplesmente rastrear áreas de interesse identificadas nos artefatos.

Analisando os requisitos R1, R2, R3 e R4 da seção 2, pode-se identificar diversas entidades, conforme ilustra a Figura 3.

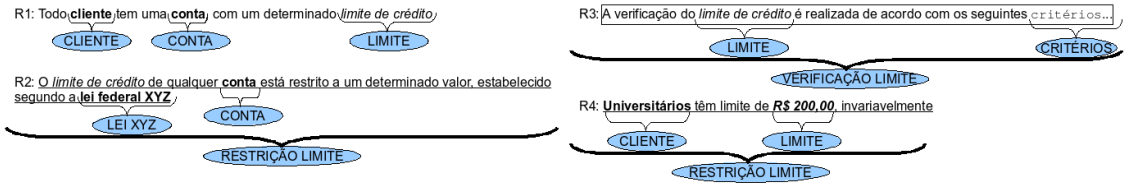


Figura 3. Entidades identificadas nos requisitos

Após identificadas as entidades de interesse, e suas origens nos requisitos ou demais artefatos, é possível derivar relacionamentos complexos semi-automaticamente, conforme descrito na seção 4.3, a seguir.

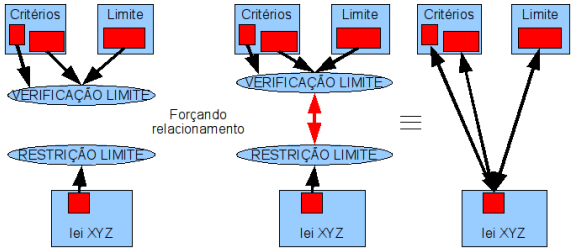
4.3. Relacionamentos

Pode-se estabelecer relacionamentos entre as entidades conforme suas características. Por exemplo, a entidade "restrição limite" é uma restrição baseada em "lei XYZ", que se aplica a "limite". Já a entidade "verificação limite" é uma ação que se baseia em "critérios" para manipular "limite". A Tabela 3 exibe os relacionamentos formados entre as entidades criadas com os requisitos R1 a R4.

Analisando as entidades identificadas na Figura 3, percebe-se que há entidades comuns a mais de um requisito (limite, conta, cliente e restrição limite). Usando estas entidades como âncoras, pode-se derivar, automaticamente, diversos relacionamentos entre os requisitos:

1. Relacionamentos diretos:
   $\{a \leftrightarrow b | a \in \{R1, R2, R3, R4\}, b \in \{R1, R2, R3, R4\}\}$**:** Relacionados entre si por originarem a entidade "limite";

Figura 4. Relacionamento forçado entre entidades



$R1 \leftrightarrow R2$**:** por tratarem da entidade "conta" ;
$R1 \leftrightarrow R4$**:** por tratarem da entidade "cliente";
$R2 \leftrightarrow R4$**:** por tratarem da entidade "restrição limite";

2. Relacionamentos indiretos:
   $\{R2 \leftrightarrow a | a \in \{R1, R2, R3, R4\}\}$**:** pois a entidade "restrição limite" restringe a entidade "limite", sendo que esta última é originada por todos os requisitos. Além disso, essa restrição se baseia em "Lei XYZ" e "Conta"(R2), e também em "Cliente" (R4);
   $\{R3 \leftrightarrow a | a \in \{R1, R2, R3, R4\}\}$**:** pois a entidade "verificação limite" afeta a entidade "limite", como no caso anterior;
   Outros: como "limite" é atributo de "conta", qualquer requisito que origine um estará indiretamente ligado a qualquer requisito que origine o outro.
3. Relacionamentos forçados: É possível estabelecer relacionamentos entre as entidades manualmente. Por exemplo, considere que a entidade "Lei XYZ" seja relacionada a "Critérios", será possível derivar novos relacionamentos automaticamente:
   $R2 \leftrightarrow R3$**:** pois R2 dá origem à entidade "Lei XYZ" e R3 à "Critérios", implicando que, em uma análise de impacto dos efeitos de alguma alteração na lei, os critérios especificados pelo requisito R3 sejam apontados para revisão. A Figura 4.3 apresenta um exemplo deste tipo de relacionamento.

Note que o crescimento dos relacionamentos derivados (entre requisitos e entre demais artefatos, tanto horizontal, quanto verticalmente) é exponencial, enquanto o crescimento dos relacionamentos manuais, entre requisitos e entidades, é linear. O modelo proposto requer o estabelecimento manual apenas de relações simples (área de interesse para entidade, ou entidade para entidade), permitindo que as relações complexas entre artefatos (áreas de interesse de um artefato para áreas de interesse de outros artefatos) sejam derivadas automaticamente.

Em relação à manutenção dos relacionamentos entre artefatos e entidades, todas as alterações realizadas sobre um artefato, que removam as áreas de interesse relacionadas a alguma entidade, farão o relacionamento desaparecer. Para que isto ocorra, o modelo exige que toda área de interesse seja identificada no próprio artefato, para que alterações neste atualizem as áreas de interesse automaticamente. Para reduzir a quantidade dos recursos e tempo necessários para o rastreamento, é necessário embutir as informações de cobertura dos requisitos no próprio software que irá atendê-los [Delgado 2006]. Por exemplo, para identificar as entidades no texto de um documento de requisitos D, pode-se usar uma sintaxe abstrata com a

forma:

$$
\begin{array}{lll}
D & ::= & palavra\ D | TAG\ D | \varepsilon \\
TAG & ::= & < ENTITY\_ID > D < \backslash ENTITY\_ID > \\
ENTITY\_ID & ::= & \mathbb{N}
\end{array}
$$

Ou seja, um documento de requisitos é formado por palavras e também por *tags*, que determinam quais são as entidades identificadas no texto. O não-terminal TAG mantém as informações das entidades identificadas no texto que cinge, identificando uma área de interesse. Na prática, este modelo requer um editor especial, que "esconda" as tags e renderize o texto entre estas. Assim, ao apagar um trecho de texto que contenha tags, o relacionamento com as entidades será simplesmente descartado. Caso uma entidade não possua relacionamento com nenhum artefato, esta será removida, bem como as relações que possui com outras entidades. A idéia por trás deste formato de definição da entidades é o analista simplesmente diga: "Estou definindo o agente X, ou parte dele", e todos os artefatos criados a partir daí sejam relacionados ao "agente X", automaticamente, conforme idealizou Cleland-Huang ([Cleland-Huang 2006]), sendo considerada pela pesquisadora a "solução ideal" para o problema da rastreabilidade.

Note que a única tarefa de manutenção da estrutura de rastreabilidade é identificar novas entidades em novas versões dos artefatos, sendo todos os relacionamentos entre artefatos derivados automaticamente. O relacionamento entre entidades e artefatos é tal que caso os artefatos mudem, as entidades e suas relações podem ser reconstruídas automaticamente. Basta comparar o estado anterior da estrutura com o atual para identificar todos os artefatos possivelmente afetados pela alteração. Isto deixa como tarefa manual apenas o rastreamento de novas entidades que porventura sejam introduzidas.

Dessa forma, todos os artefatos produzidos no ciclo de desenvolvimento podem ser utilizados para identificar entidades, basta criar um formato de representação de uma área de interesse, compatível com o modelo do artefato, como feito com o documento de requisitos D. Isto permite que a estrutura de rastreabilidade seja automaticamente atualizada à medida que os artefatos ou requisitos mudam.

4.3.1. Tipos de relação

Os relacionamentos entre artefatos podem ser tipados [Espinoza et al. 2006], dando maior semântica a estes, e facilitando a compreensão da estrutura de rastreabilidade [Ramesh and Jarke 2001]. O modelo proposto segue a mesma idéia, permitindo identificar tipos de relação entre artefatos e entidades:

defines: relação entre uma área de interesse em um artefato e uma entidade (lê-se: a área de interesse identifica a entidade);

defined_by: relação entre uma área de interesse em um artefato e um agente (lê-se: a entidade é identificada pela área de interesse);

No caso de relacionamentos entre entidades somente, são definidos os seguintes tipos de relação: *related_to:* relação de uma entidade qualquer à outra; *attribute_of:* relação de um atributo a um agente; *has_attribute:* relação de um agente a

um atributo; *restriction_of:* relação de uma restrição a um agente; *has_restriction:* relação de um agente a uma restrição; *action_of:* relação de uma ação a um agente; *has_action:* relação de um agente a uma ação; *value_of:* relação de um valor a um atributo; *has_value:* relação de um atributo a um valor; *restricted_by:* relação de uma entidade a restrição; *affected_by:* relação de uma entidade a uma ação; *based_on:* relação de uma ação ou restrição a uma entidade; *base_of:* relação de uma entidade a uma ação ou restrição; *affected_by:* relação de uma entidade a uma ação; *affects:* relação de uma ação a uma entidade; e *restricts:* relação de uma restrição a uma entidade.

Outras relações, com significado mais específico, podem ser definidas conforme desejado. A vantagem de definir tipos de relações é que as consultas na estrutura de rastreabilidade podem ser realizadas de forma mais precisa. Por exemplo, o fato de que o cliente tem uma conta (conforme R1) pode não ser relevante no processo de verificação do limite (R3). Portanto, o uso da relação *defines_agent* dificilmente seria útil na análise de impacto de uma alteração em R3. Assim, basta relizar uma consulta que exclua este tipo de relação, ou somente a relação entre R1 e R3.

4.3.2. Rastreamento horizontal

Utilizando o modelo proposto, basta relacionar suas áreas de interesse às entidades já existentes, ou a uma nova, caso necessário. Por exemplo, analisando a listagem 1, é possível relacionar áreas de interesse às entidades já identificadas, ilustradas na Figura 3. A Figura 5 apresenta estas relações.

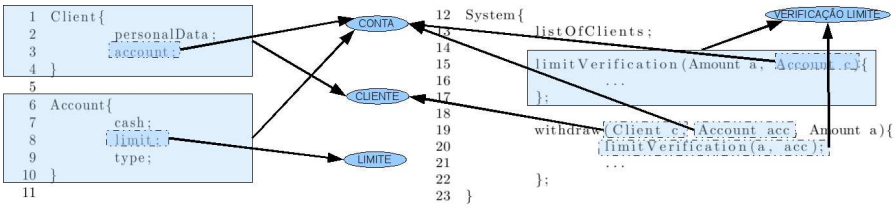


Figura 5. Relação entre áreas de interesse no código e entidades

Utilizando esta estrutura, caso um requisito seja alterado, e o analista interpretar que a ação de verificação deva ser realizada de forma diferente, todos os atributos e agentes relacionados serão identificados automaticamente como possíveis entidades afetadas. Esta identificação será propagada para os demais artefatos (possivelmente afetados) que as entidades representam.

Por exemplo, considere que a lei XYZ seja alterada e o sistema deva se adequar à nova lei. Que partes do sistema serão afetados por esta alteração? Tendo identificada a lei XYZ como uma entidade, pode-se pesquisar quais ações e restrições se baseiam em "lei XYZ". Na Tabela 3, tem-se que "restrição limite" se baseia em "lei XYZ", e conforme a Figura 4.3, a entidade "verificação limite" está relacionada a "restrição limite". Basta identificar, na implementação, as áreas de interesse relacionadas a "verificação limite". A Figura 5 identifica as linhas 15 a 17, e a linha 20, que possivelmente serão impactadas pela alteração na lei.

5. Avaliação

Com o objetivo de facilitar a avaliação do modelo proposto, foi desenvolvido um protótipo de um sistema de gerenciamento de requisitos, chamando ReMan (Requirement Manager). Por se tratar de um protótipo, este permite identificar entidades apenas em requisitos e código fonte. O modelo, implementado pelo protótipo, foi avaliado e comparado com o método tradicional de rastreamento usando matrizes de rastreabilidade. Esta avaliação foi realizada por uma equipe de desenvolvimento composta por 10 indivíduos, entre eles analistas e desenvolvedores de software de uma empresa de grande porte, em Porto Alegre - RS.

A presente proposta foi avaliada quantitativa e qualitativamente. Na avaliação quantitativa, a precisão e o esforço na abordagem por entidades foi comparada com a precisão e esforço da abordagem matricial. A precisão foi definida pela quantidade de relações criadas através destas abordages, e sua comparação com todos os relacionamentos válidos, descritos em uma tabela. Curiosamente, apesar dos participantes identificarem entidades de diferentes formas, a precisão dos relacionamentos derivados entre casos de uso e código fonte foi idêntica. O mesmo não ocorreu no rastreamento matricial, onde a precisão dos relacionamentos variou significativamente entre os participantes. A avaliação da precisão nas duas abordagens indica, pelo menos para o caso estudado, que o rastreamento utilizando entidades intermediando as relações, é muito mais preciso do que o relacionamento direto, através de matrizes: Precisão utilizando entidades (100%) > Precisão utilizando matrizes (57%). Além disso, o esforço necessário para o estabelecimento das relações, medido pelo tempo necessário para realizar o rastreamento, foi substancialmente menor ao utilizar-se entidades: Esforço utilizando entidades (12 minutos) < Esforço utilizando matrizes (33 minutos). No caso da manutenção da estrutura devido à alterações nos artefatos, o esforço foi menor ainda com o uso de entidades: Esforço utilizando entidades (0,5 minuto) < Esforço utilizando matrizes (6 minutos).

Na análise qualitativa, os participantes foram submetidos a um questionário de múltipla escolha, utilizando a escala de Lickert (variando entre 1 e 5), avaliando:

1. O grau de usabilidade da técnica no rastreamento dos artefatos: Entidades (4,5 - bom/muito bom) > Matrizes (3,4 - reguar/bom);
2. O grau de utilidade da técnica no rastreamento dos artefatos: Entidades (5 - muito bom) > Matrizes (3,2 - regular)
3. O esforço necessário para estabelecer uma estrutura de rastreabildiade: Entidades (5 - praticamente nenhum) < Matrizes (1,6 - trabalhoso/muito trabalhoso);
4. O esforço necessário para atualizar a estrutura de rastreabilidade: Entidades (5 - praticamente nenhum) < Matrizes (1,5 - trabalhoso/muito trabalhoso);
5. A viabilidade do uso da técnica no processo de desenvolvimento da empresa: Entidades (3,2 - em alguns casos) < Matrizes (2,1 - em poucos casos);
6. O atendimento da técnica em relação à proposta: Entidades (5 - em todos os casos) < Matrizes (4,5 - em todos os casos/na maioria dos casos);

Em geral, todos consideraram o modelo proposto uma alternativa viável ao rastreamento de requisitos, por não demandar esforço de interpretação na identificação de relações entre artefatos em diversos níveis de abstração. Além disso,

consideraram que o processo de rastreamento auxiliou na revisão dos requisitos, por automatizar o processo de busca dos mesmos em documentos diversos, além de facilitar a localização de inconsistências e conflitos. No entanto, o protótipo foi considerado de utilização difícil, pois não realizava o rastreamento automático do texto para as entidades que já foram identificadas pelas mesmas palavras (o que reduziria significativamente o tempo para finalizar o rastreamento). Porém, por se tratar de um problema de interface do protótipo, e não do modelo de rastreabilidade propriamente dito, ao ser comparado com o modelo matricial, foi considerado "mais barato (em termos de tempo e esforço), mais preciso e mais fácil".

Não obstante, a avaliação foi realizada sobre uma pequena parte de um sistema, com pequenos conjuntos de casos de uso e código fonte. Assim, não há certeza quanto à escalabilidade do modelo. Porém, os envolvidos na avaliação consideraram a proposta "significativamente mais viável", e acreditam que o modelo seja aplicável em projetos de grande porte, desde que exista suporte ferramental adequado, e maior automação nas atividades de identificação de entidades.

6. Trabalhos Relacionados

Tryggeseth e Nytrø [Tryggeseth and Nytro 1997] propõem que a identificação das informações de uma funcionalidade específica seja feita dinamicamente, afirmando que não há necessidade de representações persistentes, pois inibem a manutenibilidade e flexibidade. Mesmo que a geração dinâmica seja sujeita a erros, defendem que estes não inutilizam os métodos dinâmicos. O modelo apresentado incorre nos mesmos problemas citados na seção 3, pois a relação entre artefatos é realizada diretamente, e não existem informações que revelam o que estas têm em comum.

Egyed e Grunbacher [Egyed and Grunbacher 2002] utiliza o código fonte como origem de informações sobre a dependência dos artefatos, afirmando ser possível determinar a intensidade da dependência, analisando a sobreposição de classes, métodos ou linhas de código, que os artefatos relacionados possuem em comum. Desta forma, a análise do código fonte permite identificar dependências entre requisitos. No caso deste trabalho, consegue-se apenas identificar as relações entre porções de código, e não entre os demais artefatos. O rastreamento entre as porções de código dependentes e os requisitos é manual.

No trabalho de Neumuller e Grundbacher [Neumuller and Grunbacher 2006], utiliza-se identificadores em comentários, que os desenvolvedores devem adicionar ao código, de forma que possam ser associados aos requisitos, já que a utilização de convenções para nomes sempre traz o risco de falsos positivos (relações que não existem na realidade) e falsos negativos (relações existentes, que não foram identificadas). Com base nesta proposta e na anterior, decidiu-se incluir a identificação das áreas de interesse dentro dos próprios artefatos.

Maletic *et al.* [Maletic et al. 2005] acreditam que o único método prático de manter a rastreabilidade é atualizar os elos à medida que o sistema evolui. Seu trabalho traz uma representação XML entre modelos UML, que pode evoluir juntamente com os modelos do sistema. Novamente, a proposta é limitada a um pequeno conjunto de artefatos rastreáveis, e não trata do ponto mais nevrálgico da rastreabilidade: identificar as relações entre os requisitos e o restante dos artefatos.

Noll e Ribeiro [Noll and Ribeiro 2007] propõem uma modelagem ontológica sobre os requisitos, para que estes possam ser rastreados. Esta proposta tem características muito semelhantes ao modelo aqui apresentado, permitindo identificar conceitos e derivar relações entre artefatos. No entanto, está restrito ao processo de desenvolvimento RUP, e exige um especialista em modelagem ontológica. Além disso, não trata da evolução do modelo ontológico frente às alterações nos artefatos rastreados.

7. Trabalhos futuros

O modelo proposto não tem por objetivo ser completo, mas sim, extensível, para permitir o surgimento de soluções mais específicas para determinados problemas. Por exemplo, outra classificação para as entidades pode ser proposta, bem como outras formas de relacionamento e de identificação destes.

Além disso, seria interessante identificar formas de utilização do modelo proposto sobre os diversos modelos e diagramas utilizados na indústria, para representação de entidades.

Como toda proposta na área de rastreabilidade, o suporte ferramental é considerado essencial. Espera-se que implementações aplicáveis no mundo real sejam desenvolvidas a partir do modelo proposto. O protótipo ReMan já desenvolvido é apenas uma prova-de-conceito, mas que provavelmente pode ser utilizado como base na compreensão e melhoria do modelo aqui apresentado.

Em especial, devem ser realizadas pesquisas na área de recuperação automática de rastreabilidade, que possibilitem a criação automática de entidades. Acredita-se que a combinação do método proposto com as técnicas de recuperação já desenvolvidas possam facilitar ainda mais o processo de criação de elos. Esta idéia é baseada na hipótese de que relações simples, entre artefatos e entidades, possam ser criadas com grande precisão através do uso de tais técnicas.

Por fim, a semântica dos modelos, utilizados para representar os diferentes pontos de vista sobre o sistema, poderia ser aproveitada na determinação dos elos. Isto permitiria consultas mais eficientes na estrutura e descoberta de relações com muito mais precisão.

8. Conclusão

Com o intuito de mudar a realidade de alto custo na manutenção da rastreabilidade, e fornecer uma solução para os problemas inerentes ao uso de relações diretas, este trabalho apresenta uma inovação sobre os tradicionais métodos: a concretização do raciocínio por trás da interpretação dos requisitos em um modelo mais detalhado que o matricial, através de entidades identificadas nos artefatos. Este é utilizado para representar os demais artefatos produzidos no ciclo de vida do sistema, rastrear os requisitos a estes, e identificar relacionamentos entre todos os artefatos automaticamente. Assim, a manutenção dos elos de rastreabilidade já existentes torna-se automática, salvo casos de inclusão de novos elos, devido à semântica que o modelo fornece, permitindo derivar relações.

Para avaliar a aplicabilidade do modelo, o protótipo ReMan foi implementado, adaptando o conceito à orientação a objetos. Conforme as avaliações, alte-

rações nos requisitos e no código realmente permite que os elos existentes sejam atualizados semi-automaticamente, mantendo consistentes as relações entre requisitos e implementação.

Espera-se que este trabalho sirva como base para diversas novas idéias e conceitos, na ainda problemática área da rastreabilidade de requisitos. Acredita-se que o modelo apresentado serve como um passo à frente nas pesquisas da área, para que um dia faça-se referência não ao "problema da rastreabilidade", mas sim à "solução da rastreabilidade".

Referências

Cleland-Huang, J. (2006). Requirements traceability - when and how does it deliver more than it costs? In *RE '06: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, page 323, Washington, DC, USA. IEEE Computer Society.

Cleland-Huang, J., Chang, C. K., and Christensen, M. (2003). Event-based traceability for managing evolutionary change. *IEEE Trans. Softw. Eng.*, 29(9):796–810.

Cleland-Huang, J., Chang, C. K., Sethi, G., Javvaji, K., Hu, H., and Xia, J. (2002). Automating speculative queries through event-based requirements traceability. In *RE '02: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, pages 289–298, Washington, DC, USA. IEEE Computer Society.

Cleland-Huang, J., Zemont, G., and Lukasik, W. (2004). A heterogeneous solution for improving the return on investment of requirements traceability. In *RE '04: Proceedings of the Requirements Engineering Conference, 12th IEEE International (RE'04)*, pages 230–239, Washington, DC, USA. IEEE Computer Society.

Delgado, S. (2006). Next-generation techniques for tracking design requirements coverage in automatic test software development. In *Systems Readiness Technology Conference, IEEE*, pages 806–812, Washington, DC, USA. IEEE Computer Society.

Egyed, A. and Grunbacher, P. (2002). Automating requirements traceability: Beyond the record & replay paradigm. In *ASE '02: Proceedings of the 17th IEEE international conference on Automated software engineering*, page 163, Washington, DC, USA. IEEE Computer Society.

Espinoza, A., Alarcon, P. P., and Garbajosa, J. (2006). Analyzing and systematizing current traceability schemas. *sew*, 0:21–32.

Hayes, J. H. and Dekhtyar, A. (2005). Humans in the traceability loop: can't live with 'em, can't live without 'em. In *TEFSE '05: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pages 20–23, New York, NY, USA. ACM Press.

Heindl, M. and Biffl, S. (2006). Risk management with enhanced tracing of requirements rationale in highly distributed projects. In *GSD '06: Proceedings of the 2006 international workshop on Global software development for the practitioner*, pages 20–26, New York, NY, USA. ACM Press.

Kelleher, J. and Simonsson, M. (2006). Utilizing use case classes for requirement and traceability modeling. In *MS'06: Proceedings of the 17th IASTED international conference on Modelling and simulation*, pages 617–625, Anaheim, CA, USA. ACTA Press.

Lucia, A. D., Fasano, F., Oliveto, R., and Tortora, G. (2006). Can information retrieval techniques effectively support traceability link recovery? In *ICPC '06: Proceedings of the 14th IEEE International Conference on Program Comprehension (ICPC'06)*, pages 307–316, Washington, DC, USA. IEEE Computer Society.

Maletic, J. I., Collard, M. L., and Simoes, B. (2005). An xml based approach to support the evolution of model-to-model traceability links. In *TEFSE '05: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pages 67–72, New York, NY, USA. ACM Press.

Marcus, A., Xie, X., and Poshyvanyk, D. (2005). When and how to visualize traceability links? In *TEFSE '05: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pages 56–61, New York, NY, USA. ACM Press.

Murta, L. G. P., van der Hoek, A., and Werner, C. M. L. (2006). Archtrace: Policy-based support for managing evolving architecture-to-implementation traceability links. In *ASE '06: Proceedings of the 21st IEEE International Conference on Automated Software Engineering (ASE'06)*, pages 135–144, Washington, DC, USA. IEEE Computer Society.

Neumuller, C. and Grunbacher, P. (2006). Automating software traceability in very small companies: A case study and lessons learne. In *ASE '06: Proceedings of the 21st IEEE International Conference on Automated Software Engineering (ASE'06)*, pages 145–156, Washington, DC, USA. IEEE Computer Society.

Noll, R. P. and Ribeiro, M. B. (2007). Ontological traceability over the unified process. In *ECBS '07: Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 249–255, Washington, DC, USA. IEEE Computer Society.

Penta, M. D., Gradara, S., and Antoniol, G. (2002). Traceability recovery in rad software systems. In *IWPC '02: Proceedings of the 10th International Workshop on Program Comprehension*, page 207, Washington, DC, USA. IEEE Computer Society.

Ramesh, B. and Jarke, M. (2001). Toward reference models for requirements traceability. *IEEE Trans. Softw. Eng.*, 27(1):58–93.

Tryggeseth, E. and Nytro, O. (1997). Dynamic traceability links supported by a system architecture description. In *ICSM '97: Proceedings of the International Conference on Software Maintenance*, pages 180–187, Washington, DC, USA. IEEE Computer Society.

yi Jiang, H., Nguyen, T. N., Chang, C. K., and Dong, F. (2007). Traceability link evolution management with incremental latent semantic indexing. *31st Annual International Computer Software and Applications Conference(COMPSAC 2007)*.