# Uma Proposta de Modelagem para a Generalização de Elos de Rastreabilidade

Elias Canhadas Genvigir[1] [2]
Nandamudi Lankalapalli Vijaykumar[1]

**Resumo:** Diversos modelos propõem tipos pré–definidos de elos para a rastreabilidade de requisitos. Tais modelos fazem extensivas observações sobre as práticas da rastreabilidade, mas são limitados tanto pelos tipos de elos pré–definidos quanto pela capacidade de incluir atributos para os elos. Este trabalho propõe um modelo para rastreabilidade de requisitos que generaliza os tipos de elos já definidos, permitindo a adição de novos padrões e a inclusão de atributos para os elos que serão utilizados em um determinado processo de rastreabilidade.

**Abstract:** Several models proposed traceability links that provide pre–defined groups of links for requirements traceability. These models are limited to pre–defined links without the ability to add new attributes to the existing links. This work proposes a model for requirements traceability that generalizes the types of links already established in the literature and enables addition of new standards allowing the inclusion of attributes to the links that will be used in a specific traceability process.

## 1 Introdução

Um software é dirigido à realização de tarefas, que estão associadas à solução computacional de um problema do domínio da natureza humana. Dada a amplitude dos problemas aos quais o software pode ser dirigido é necessário que sua produção atinja padrões básicos de qualidade e produtividade.

Com a evolução da pesquisa e a necessidade de melhoria na produção de software a Engenharia de Software foi se especializando em várias subáreas. Uma classificação para essas subáreas foi realizada pelo projeto *Guide to the Software Engineering Body of Knowledge* – SWEBOK [1] que definiu dez áreas de conhecimento para a Engenharia de Software, sendo que a primeira área de conhecimento definida trata sobre os requisitos de software.

Os requisitos são de extrema importância, pois são definidos durante os estágios iniciais do desenvolvimento, como uma especificação do que deve ser implementado descrevendo

[1]Universidade Tecnológica Federal do Paraná – UTFPR. Av. Alberto Carazzai, 1640, CEP: 86300–000, Cornélio Procópio, PR, Brasil –{elias@utfpr.edu.br}
[2]Instituto Nacional de Pesquisas Espaciais – INPE. Av. dos Astronautas, 1.758, CEP: 12227–010, São José dos Campos, SP, Brasil –{elias,vijay@lac.inpe.br}

como o sistema deve comportar–se, detalhando os atributos ou propriedades do sistema, ou ainda, podendo estabelecer restrições sobre o processo de desenvolvimento [2]. A importância dos requisitos para o desenvolvimento de sistemas é tão crítica que nenhuma outra parte do trabalho com software incapacita e prejudica tanto o sistema e, depois de concluído, nenhuma outra parte é mais difícil de corrigir do que os requisitos [3].

A área da Engenharia de Software que trata sobre os requisitos de software é conhecida como Engenharia de Requisitos, que é o processo de descoberta dos requisitos, de identificação dos envolvidos e suas necessidades e de documentação de forma que seja útil para a análise, comunicação, e a subseqüente implementação [4].

Entre as várias atividades que compõem a Engenharia de Requisitos destaca–se a Rastreabilidade, que é comumente usada para descrever a referência para um grupo coletivo de requisitos baseados em seus relacionamentos.

Os relacionamentos são estabelecidos entre requisitos e artefatos de software usando um elo. Elos são elementos necessários para estabelecer a rastreabilidade, enquanto que artefatos são considerados informações produzidas ou modificadas como parte de um processo de Engenharia de Software [5]. Artefatos podem ser modelos, documentos, código fonte, seqüências de testes, requisitos ou executáveis. Esses são os elementos de um sistema que podem ser rastreados [6].

Existe uma série de propostas para modelos de elos de rastreabilidade, também definidos como modelos de referência[5], que provêm padrões pré–definidos de grupos de elos [5, 7, 8, 9]. Tais pesquisas fazem extensivas observações das práticas da rastreabilidade, mas seus modelos são limitados na fixação dos tipos de elos, ou seja, os grupos de elos são definidos para atingir uma determinada solução para o domínio do problema para o qual o modelo é proposto, o que pode limitar as práticas da rastreabilidade. Em adição, esses modelos não permitem a inclusão de atributos ou semânticas ricas[10, 11] o que restringe a descrição dos elos.

Modelos de referência podem ser muito úteis, visto que grande é o esforço utilizado para analisar o domínio para um novo sistema. Nos casos de domínios padronizados é reportado que o uso de modelos de referência podem reduzir o esforço em até 80% [12]. Entretanto em domínios, não suficientemente padronizados, os modelos de referência podem limitar o desenvolvimento da aplicação.

Este trabalho aborda modelos de referência para rastreabilidade e suas limitações, focando na definição de um modelo que generaliza a definição dos elos para diferentes processos de rastreabilidade. São abordados modelos que representam elos de rastreabilidade e com base nas limitações já mencionadas é proposto um modelo que permite: (i) representar vários tipos de elos já suportados pelos modelos existentes; e (ii) criar novos tipos de elos para servir a uma necessidade específica de um projeto. O modelo apresentado aqui pode

também adicionar atributos para melhorar a semântica dos elos.

O modelo proposto tem por objetivo permitir a criação de diferentes tipos de elos pelos desenvolvedores, além da inserção de atributos a esses elementos, possibilitando que os desenvolvedores possam definir padrões de elos, conforme as necessidades específicas de seus projetos, viabilizando um maior nível de detalhes sobre a rastreabilidade.

A pesquisa é organizada da seguinte forma: a seção 2 apresenta os conceitos sobre rastreabilidade, seus aspectos e importância e sobre elos e seus tipos. Na seqüência é explorado, brevemente, o conceito de modelos, necessário para compreender o uso de modelos para rastreabilidade e então é apresentado o modelo proposto, seguido de sua modelagem e demonstração.

## 2 Rastreabilidade

A rastreabilidade está intimamente associada ao processo de produção de artefatos de software, especificamente aos requisitos e a capacidade de estabelecer vínculos entre esses requisitos e outros artefatos que os satisfaçam. Essa característica é observada por Letelier [7] que afirma que a rastreabilidade de requisitos ajuda a garantir uma contínua concordância entre os requisitos dos interessados no sistema e os artefatos produzidos ao longo do processo de desenvolvimento de software. Na mesma linha Palmer [13] aponta que a rastreabilidade dá a assistência essencial na compreensão do relacionamento que existe com e entre requisitos de software, projeto e implementação sendo que estes relacionamentos auxiliam o projetista a mostrar quais elementos do projeto satisfazem os requisitos.

Sobre as vantagens da rastreabilidade é observado que seu uso ajuda a estimar variações em cronogramas e em custos do projeto [13, 14, 15, 16]. Além disso, Sayão e Leite [17] apontam que a rastreabilidade pode auxiliar gerentes de projeto a: verificar a alocação de requisitos a componentes de software; resolver conflitos entre requisitos; verificar requisitos nos processos de testes; corrigir defeitos através da identificação do componente de origem do erro; validar o sistema junto aos clientes; analisar o impacto na evolução dos sistemas; prever custos e prazos; e gerenciar riscos e reuso de componentes.

Dependendo de sua semântica, a rastreabilidade pode ser usada para (a) assistir o processo de verificação dos requisitos para um sistema, (b) estabelecer o impacto de mudanças na especificação de requisitos através de seus artefatos ou da documentação (Ex. projeto, teste e implementação de artefatos), (c) compreender a evolução de um artefato, e (d) compreender os aspectos do projeto e o suporte de *Rationales*. Assim, a geração e a manutenção de tais relações podem fornecer uma base mais eficaz para a garantia da qualidade do sistema, a gerência das mudanças, e a manutenção do software [18].

Apesar de ser usada por vários processos ainda é evidente a falta de padrões para a

qualidade da rastreabilidade, isso pode ser observado nas normas e modelos como o CMMI–DEV [19] que aponta a prática específica: 1.4 Manter a Rastreabilidade Bidirecional para os Requisitos, ou a norma ISO/IEC 15504 –2 [20] que possui as práticas base: ENG.3.4 – Estabelecer a rastreabilidade e PRO.4.5 – Manter a rastreabilidade, em ambos casos propõem–se apenas que a rastreabilidade seja executada mas nenhum padrão para a qualidade da rastreabilidade é estabelecido.

Um ponto fundamental para a qualidade trata sobre as métricas, mas existem poucas pesquisas sobre esse tema aplicado à rastreabilidade. Algumas métricas são propostas por Costello e Liu [21] que definem cinco tipos: cobertura de próximo nível (COV), profundidade plena e alta cobertura (DHCOV), vinculação estatística, rastreabilidade inconsistente (ITM), e rastreabilidade indefinida (UTM); e por Hull et al. [22] que definem cinco elementos para a mensuração da rastreabilidade: amplitude, profundidade, crescimento, balanço e mudança latente.

Sobre os tipos de rastreabilidade basicamente existem dois tipos, a pré–rastreabilidade que está concentrada no ciclo de vida dos requisitos antes de serem incluídos na especificação de requisitos, e a pós–rastreabilidade, que está concentrada no ciclo de vida dos requisitos após serem incluídos na especificação [23].

A capacidade de rastrear um requisito até seus refinamentos é definida como rastrear para frente (*Forwards*), e a capacidade de rastrear um refinamento até sua origem é definida como rastrear para trás (*Backwards*) [14]. Os dois tipos são essenciais e fazem parte tanto da Pós quanto da pré–rastreabilidade.

### 2.1 Elos de Rastreabilidade

Como já afirmado a rastreabilidade é realizada entre requisitos e demais artefatos fazendo uso de elos, esses elementos podem ser formalizados como: $A = \{a_1, a_2, a_3, ..., a_n\}$ representa todos os artefatos produzidos e identificados no processo de desenvolvimento, então $E = \{e_1, e_2, e_3, ..., e_n\} \subset A$ é um subgrupo dos artefatos que representam elos, e $R = \{r_1, r_2, r_3, ..., r_n\} \subset A$ é um subgrupo de $A$ que representam os requisitos.

Várias propostas têm sido feitas para elos de rastreabilidade assim como para modelos que suportem: (i) algum padrão focado na manutenção do processo e (ii) a representação desses elos.

Basicamente os elos estão associados à uma metodologia ou à alguma informação do domínio do processo de desenvolvimento. Neste aspecto, elos foram criados para orientação a objetos [18], orientação a aspectos [24], desenvolvimento centrado em visão [25], e desenvolvimento dirigido a modelo [26, 27]. Além disso elos podem ser criados por aspectos ambientais e organizacionais [5]; no tipo de informação a ser rastreada [9]; ou ainda por con-

figurações que integram especificações textuais [7]. O principal problema desses elos é que eles são criados e usados somente para um propósito específico.

No caso do processo de engenharia de requisitos os elos são usados nas atividades de: validação, análise, evolução e referência cruzada entre requisitos e artefatos [13]. Também fazem uso desse recurso os processos de gerência de projeto (auxiliar a predição de custo, restrições ou impactos); de manutenção; de teste (geração de casos de teste baseados em cenários ou modelos); e no processo da garantia da qualidade (validação do software)[6, 8, 13, 23, 28, 29, 30].

Um elo representa explicitamente o relacionamento definido entre dois artefatos $a_1$ e $a_2$. $a_1 \rightarrow a_2$ são considerados como diretamente ligados enquanto um elo indireto usa um artefato intermediário como em $a_1 \rightarrow a_2 \rightarrow a_3$ [6]. Um elo é estabelecido entre um artefato origem e um artefato destino, Figura 1.


**Figura 1.** Elementos básicos de um elo.

Informações sobre a origem e o destino são suficientes para suportar a rastreabilidade para frente e para trás, mas outras atividades da engenharia de requisitos, como o auxílio à previsão de custos e prazos e a resolução de conflitos, necessitam de outros elementos para sua execução [10, 11].

Várias categorias de elos podem ser determinadas usando como base os atributos e propriedades ou a aplicação desses elos no processo de desenvolvimento. Devido a essas características existem muitos tipos de elos descritos na literatura. A Tabela 1 mostra alguns desses elos.

**Tabela 1.** Exemplos de elos descritos na literatura classificados por seus respectivos autores e subgrupos.

| **Autor** | **Grupos** | **Tipos de elo** |
|---|---|---|
| Ramesh e Jarke[5] | Relacionado ao produto | Satisfação |
| | | Dependência |
| | Relacionado ao processo | Evolução |
| | | *Rationales* |
| Toranzo e Castro [9] | | Satisfação |
| | | Recurso |
| | | Responsabilidade |
| | | Representação |
| | | Alocação |
| | | Agregação |
| Pohl [31] | Condição | Restrições |
| | | Pré–condições |
| | Documentos | Exemplos |
| | | Propósito |
| | | Caso de teste |
| | | Comentários |
| | | Segundo Plano |
| | Abstração | Refinado |
| | | Generalizado |
| | Evolução | Elaborado |
| | | Formalizado |
| | | Baseado em |
| | | Satisfação |
| | | Substituído |
| | Conteúdo | Similar |
| | | Comparação |
| | | Contradição |
| | | Conflito |
| De Lucia et al. [32] | | Dependência |
| | | Composição |
| | | Indireto |
| Spanoudakis et al. [18] | | Sobreposição |
| | | Execução_Requerida |
| | | Característica_Requerida |
| | | Parcialmente_Realizável |
| Aizenbud–Reshef et al.[26] | | Imposto |
| | | Inferido |
| | | Manual |
| | Computacional | Derivação |
| | | Análise |

## 2.2 Técnicas de Rastreabilidade

O desenvolvimento e o uso de técnicas de rastreabilidade tiveram início na década de 1970 [26] e o primeiro método usado para expressar e manter a rastreabilidade foi a referência

cruzada [33].

Outras técnicas podem ser usadas tanto para representar como para estabelecer relacionamentos incluindo matrizes [14, 34]; dependência de frases chaves [35]; integração de documentos [36]; hipertexto [37, 38, 39], grafos [8]; métodos formais [40]; esquemas dinâmicos [6, 41, 42]; sistemas baseados em suposição da manutenção de verdade [43, 44]; e redes de confiança [45].

Técnicas automatizadas ou semi automatizadas de rastreabilidade também foram desenvolvidas, como a detecção de existência de elos entre documentos de requisitos, documentos de projeto, e código fonte usando uma máquina de aprendizagem [46]; técnicas de análise altamente escaláveis para análise automática de consistência entre requisitos e projetos [47]; técnicas para recuperação de informação – RI (*Information Retrieval*) [48, 49, 50] como a indexação por análise da semântica latente [51, 52] que são usadas para recuperar elos de rastreabilidade ou na recuperação por construção de elos sobre a formalização de semânticas [53].

Pesquisas mais recentes vêm mostrando que as técnicas baseadas em RI podem ser usadas para descobrir dinamicamente elos [41, 54, 55, 56], além de permitir o relacionamento entre vários tipos de artefatos [53] como código e documentação [55, 57], requisitos e projeto [54], artefatos e requisitos [58]. Outra característica dessas técnicas é que elas podem prover a rastreabilidade sem a necessidade de manter armazenados os elos.

A técnica conhecida como modelo de reflexão de software [59] checa a conformidade da implementação do código com o modelo de projeto usando regras de mapeamento entre o modelo de projeto e o modelo de origem extraído para o código de implementação. Outro trabalho nesta mesma linha é o de Richardson e Green [60], que usa uma síntese automatizada para inferir elos. Perturbações no artefato de origem são analisadas e usadas para identificar os elos entre os artefatos de origem e destino.

Além da rastreabilidade entre artefatos algumas técnicas como a rastreabilidade baseada em eventos [6] podem ser usadas para rastrear requisitos associados a desempenho em modelos de desempenho executáveis [6, 61, 62], e também para rastrear a qualidade de requisitos implementados através de padrões de projeto já conhecidos [63], que podem ser ligados usando seus invariantes [64].

### 2.3 Modelos para rastreabilidade

Segundo Mellor et al. [65] um modelo é um grupo coerente de elementos que descrevem algo construído através de alguma forma de análise para algum propósito particular, podendo ser expresso por uma linguagem (tanto textual ou gráfica) que por sua vez possui certo grau de abstração.

No caso da rastreabilidade, os modelos são desenvolvidos com base nas informações sobre um determinado domínio da rastreabilidade (usuários, práticas, metodologias, normas, etc.). Assim a definição dos elos está associada com os conceitos modelados.

Ramesh e Jarke [5] criaram um meta–modelo para rastreabilidade utilizando como base uma extensa pesquisa empírica sobre as conseqüências dos diferentes usos, perspectivas e necessidades de informação dos envolvidos no processo de desenvolvimento.

Os autores encontraram três categorias de informações associadas à rastreabilidade: interessados; fontes (padrões, normas, regras institucionais, etc); e objetos (objetos conceituais, modelos, ou artefatos), que foram utilizadas para estabelecer o meta-modelo.

Além das três classes de informação a pesquisa permitiu aos autores criar uma classificação dos usuários da rastreabilidade. Esses foram divididos, quanto suas práticas, em dois grupos distintos: usuários sofisticados e normais (*high–end e low–end users*). Com base nessa classificação dois modelos foram criados para representar seus processos de rastreabilidade.

Adicionalmente à criação do meta–modelo, conceitualmente simples mas muito amplo em seus elementos e elos, o trabalho de Ramesh e Jarke [5] apresenta uma classificação muito interessante sobre os usuários da rastreabilidade, dando grande valor às práticas utilizadas por estes nas organizações. Entretanto, esse modelo também pré–estabelece os tipos de elos a serem utilizados, permitindo apenas a divisão das classes pré–estabelecidas. Outro problema encontrado no modelo é que este não possibilita a adição de atributos aos elos, o que permitiria o enriquecimento da semântica desses artefatos, apesar da existência do tipo de elo *Rationale* que é importante para determinar as razões da criação ou alterações de artefatos.

O modelo de Toranzo e Castro [9] faz o uso de representação gráfica através de diagramas de classes da UML. Esse modelo tem como conceito a classificação das informações a serem rastreadas, que são divididas em quatro classes: Ambientais, Organizacionais, Gerenciais e de Desenvolvimento. Com base na classificação os autores desenvolveram um meta–modelo; um modelo intermediário e um processo para aplicação das estratégias anteriores sobre a rastreabilidade. É dada atenção a aspectos gerenciais, como a definição do elo do tipo responsabilidade. Esse aspecto é notado pela influência dos contextos da classificação definida (Ambientais, Organizacionais, Gerenciais e de Desenvolvimento). Entretanto, os tipos de elos também são pré–definidos sendo focados apenas nos contextos utilizados no modelo, devido a essa característica outros padrões de gerência ou de rastreabilidade, como dirigida à visão ou a modelos poderão não ser completamente expressos.

Uma discussão sobre o problema da pré–especificação de elos nos modelos de rastreabilidade é apresentada por Aizenbud–Reshef et al. [26]. Os autores propõem uma solução, para este problema, usando o conceito de *Model Driven Architecture* – MDA. Esta proposta considera que a criação de modelos para sistemas deve incluir um meta-modelo de requisi-

tos, sendo que os modelos resultantes, que incluem um modelo explícito de requisitos, podem prover a rastreabilidade entre requisitos e outros artefatos. A existência de tais modelos não provê mecanismos de como produzir, automaticamente, os elos entre requisitos e suas dependências, ou seja, segundo os autores o fato de se possuir uma semântica para produzir elos significativos não garante uma efetiva estratégia de rastreabilidade. Assim os elos necessitam ser construídos e mantidos manualmente.

Segundo Aizenbud–Reshef et al. [26] o ideal seria gerar os artefatos, ou suas estruturas básicas, e produzir os elos entre eles, e se necessário os elos seriam detalhados. Para tanto a MDA poderia ser uma alternativa para realizar essas tarefas.

Apesar da crítica sobre os modelos de rastreabilidade, que pré–definem os tipos de elos, a proposta de Aizenbud–Reshef et al. está vinculada a uma solução baseada em arquitetura, o que pode ser um problema para processos de desenvolvimento baseados em outros conceitos. Entretanto, esta proposta pode ser muito relevante para os envolvidos com a MDA.

Os modelos existentes provêm padrões pré–definidos, de grupos de elos, que podem ser suportados pelas soluções conceituais propostas por esses modelos. Esses trabalhos fazem extensivas observações das práticas da rastreabilidade, mas seus modelos são limitados na fixação dos tipos de elos, ou seja, são definidos para cobrir uma determinada abordagem de solução para o domínio do problema ao qual seus modelos são propostos [26].

Outro problema vigente na área abrange a restrição da descrição dos elos, sendo que estes, na maioria dos casos, representam apenas a ligação entre dois elementos do processo de desenvolvimento, ou seja, esses modelos não permitem a inclusão de atributos ou semânticas ricas, assim informações necessárias para estabelecer análises, como avaliação de impacto, derivação ou cobertura [10, 11, 22], sobre a rastreabilidade, não são apresentadas, representadas ou mesmo suportadas por esses modelos.

## 3 Modelo Proposto

Os modelos apresentados são limitados no que diz respeito aos tipos de elos, ou seja, são fixados, e conseqüentemente são definidos para lidar apenas com soluções específicas para um domínio fixo de problemas.

Com base nas restrições dos modelos já apresentados é proposto um modelo baseado em quatro elementos que podem representar vários tipos de artefatos. Diferente das outras abordagens o modelo proposto permite que sejam definidos tanto os padrões dos requisitos, dos artefatos ou dos elos a serem utilizados ao longo do projeto. A Figura 2 apresenta os quatro elementos que compõem o modelo.

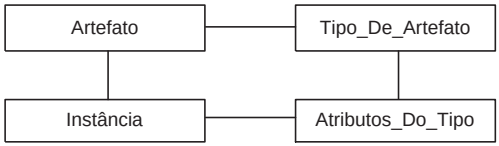


**Figura 2.** Elementos do modelo proposto para a rastreabilidade de requisitos.

O modelo se baseia na generalização de todos os tipos de artefatos que possam fazer parte do processo de rastreabilidade, focando a representação das informações. Essa característica permite tanto a criação de novos tipos de elos quanto de outros artefatos, conforme a necessidade dos envolvidos na rastreabilidade. Dois casos particulares justificam essa generalização. O primeiro trata sobre a agregação de novos campos para um dado tipo de elo ou artefato. O segundo trata sobre a inserção de novos tipos de artefatos.

Para o primeiro caso os elos constituem, na maioria das aplicações existentes, apenas a existência da ligação entre dois artefatos, sendo que informações importantes não são agregadas a essa ligação. Ao constituir um elo de rastreabilidade deve–se permitir que sejam inseridas informações do processo ou adjacentes a esse, como informações sobre qualidade e *Rationale*.

O segundo caso consiste da necessidade de inclusão de novos tipos de elos ou outros artefatos ao longo do projeto ou para um novo projeto. Permitir a definição dos tipos de elos somente no início do projeto não condiz com a necessidade de evolução e adaptação do projeto frente às alterações que possam vir a ser realizadas.

A relação entre os elementos do modelo pode ser visualizada através de um diagrama entidade relacionamento, Figura 3.

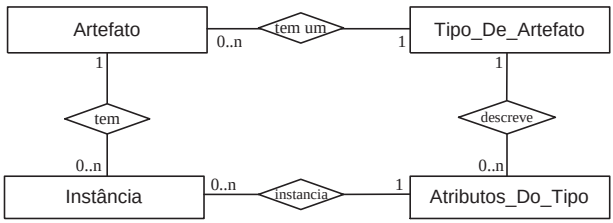


**Figura 3.** Diagrama Entidade Relacionamento do modelo proposto.

No modelo relacional um esquema de relação é dado por $P(A, K)$, onde $P$ é o nome dado à relação e $A = \{A_1, A_2, ..., A_n\}$ é um finito grupo de atributos. Cada atributo $A_i$ representa um papel desempenhado por algum domínio $D$, dom($A_i$). O número $n$ de atributos

define o grau da relação. Enquanto que $K$ consiste de um grupo finito de atributos chaves $K = \{K_1, K_2, ..., K_n\}$, onde $K \subseteq A$. Dizer que $K = \{K_1, K_2, ..., K_n\}$ é a chave de um esquema de relação é dizer que qualquer relação válida $p$ em $P$ tem a propriedade que para qualquer tupla distinta $t_1$ e $t_2$ em $p$, $t_l(K) \neq t_2(K)$, e que nenhum outro subgrupo de $K$ tem essa propriedade.

Assim, como no modelo relacional, o modelo proposto é definido como um grupo $S$ formado por quatro esquemas de relação *S=(Tipo_De_Artefato (TF), Atributos_Do_Tipo (AT), Instancias (I), Artefatos (A))*. A Tabela 2 mostra as relações e seus atributos.

**Tabela 2.** Descrição dos elementos do Modelos.

| **Nome da Relação** | **Nomes dos Atributos** | **Domínio** |
|---|---|---|
| Tipo_De_Artefato | TipoDeArtefatoID | chave primária |
| | Nome | nomes de artefatos |
| | Descrição | descrição dos artefatos |
| Atributos_Do_Tipo | AtributosDoTipoID | chave primária |
| | TipoDeArtefatoFK | chave estrangeira para $TF$ |
| | Name | nomes dos atributos da relação $TF$ |
| | Descrição | descrição dos atributos |
| Instância | InstânciaID | chave primária |
| | ArtefatoFK | chave estrangeira da relação $A$ |
| | AtributosDoTipoFK | chave estrangeira da $AT$ |
| | Valor | valores para os atributos do artefato |
| Artefato | ArtefatoID | chave primária |
| | TipoDeArtefatoFK | chave estrangeira de $TF$ |
| | Data | data da criação do artefato |
| | Versão | versão do artefato |
| | Descrição | descrição do artefato |

Cada elemento de $S$ pode ser instanciado. Por exemplo, *Artefato* pode ser instanciado como $a_i(A)$ tendo grau 5 (domínios: ArtefatoID, TipoDeArtefatoFK, Data, Versão, e Descrição), exemplo:

$$a_i(A) \subseteq (dom(ArtefatoID) \times dom(TipoDeArtefatoFK) \times dom(Data) \times dom(\textit{Versão}) \times dom(\textit{Descrição}))$$

O mesmo conceito se aplica aos outros elementos de $S$.

Para criar um artefato $a_i$, no modelo proposto, são necessários os seguintes passos:

1. Definir o tipo de artefato – Especificar o nome e a descrição de uma categoria na qual uma instância de artefato fará parte, como: código fonte, documento, requisito, elo,

caso de uso, etc.

2. Determinar os atributos do tipo do artefato – Os atributos são usados para detalhar o tipo de artefato. Por exemplo, um tipo de artefato caso de uso pode possuir os seguintes atributos: nome, ator, pré–condições, fluxo principal, etc.

3. Criar um artefato – A criação de um artefato depende do tipo ao qual o novo artefato será instanciado e do domínio de valores que serão atribuídos a ele. Suas características são definidas pela relação *Atributos_Do_Tipo*, assim através das chaves dessa relação é possível associar os atributos.

Os valores dos atributos de $a_i$ são designados na relação *Instância*, que através de suas restrições estabelece que esses valores pertencem exclusivamente ao artefato $a_i$.

O processo de criação de um artefato pode ser visualizado através de um diagrama de atividades usando a notação do *Software Process Engineering Metamodel Specification*– SPEM [66, 67] proposto pelo Object Management Group – OMG, Figura 4.

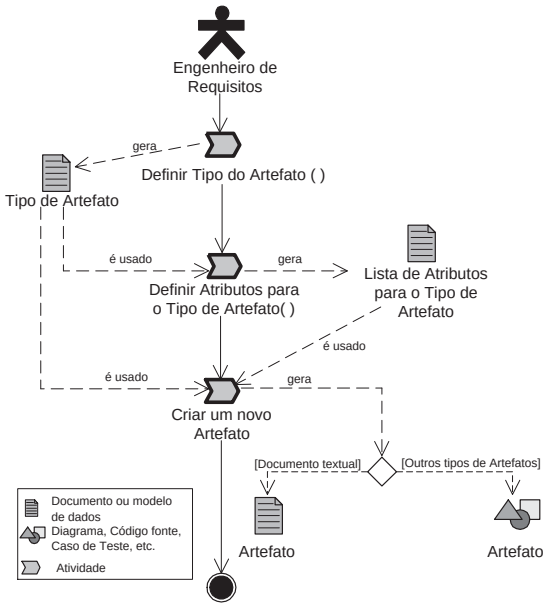


**Figura 4.** Diagrama de atividades do processo de criação de um artefato usando o modelo.

O artefato $a_i$ é definido pelos atributos da relação *Artefato*, pelo tipo definido na rela-

ção *Tipo_Do_Artefato*,e pelos valores armazenados na relação *Instância*.

Na Figura 5 um artefato $a_i$ é definido como sendo do tipo Requisito Funcional e tendo dois atributos (Descrição e Fator de Risco). Cada um desses atributos tem suas instâncias ("O sistema deve..." e "Risco1") que estão associados ao artefato $a_i$.

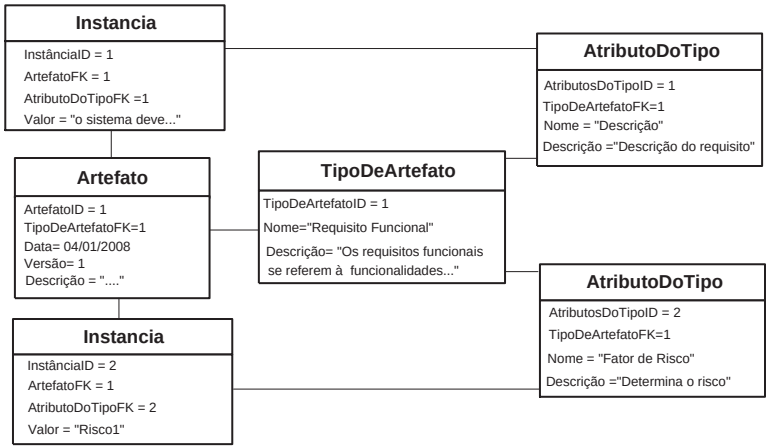


**Figura 5.** Valores do artefato $a_1$.

A visualização do artefato $a_1$ pode ser feita através de três operações conhecidas na álgebra relacional: projeção($\pi$), seleção($\sigma$) e junção natural ($\bowtie$):

$$\pi_{a.ArtefatoID,tf.Nome,at.Nome,i.Valor}(\sigma_{a.ArtefatoID=1}(a \bowtie i \bowtie tf \bowtie at). \qquad (1)$$

A Tabela 3 mostra o resultado da expressão 1.

**Tabela 3.** Resultado da expressão 1.

| a.ArtefatoID | tf.Nome | at.Nome | i.Valor |
|---|---|---|---|
| 1 | Requisito Funcional | Descrição | O sistema deve... |
| 1 | Requisito Funcional | Fator de Risco | Risco1 |

Todos os atributos necessários para documentar um determinado artefato podem ser criados através da relação *at*. Para estabelecer um elo é necessário considerar a criação de pelo menos mais um artefato, dito $a_2$. Aqui $a_2$ é do tipo *tf*="Caso de Uso", que possui dois atributos *at*="Identificador" e *at*="Nome", que têm valores instanciados *i.valor* ="2" e *i.valor* ="Controlar dados do cliente". O artefato $a_2$ é apresentado na Figura 6.

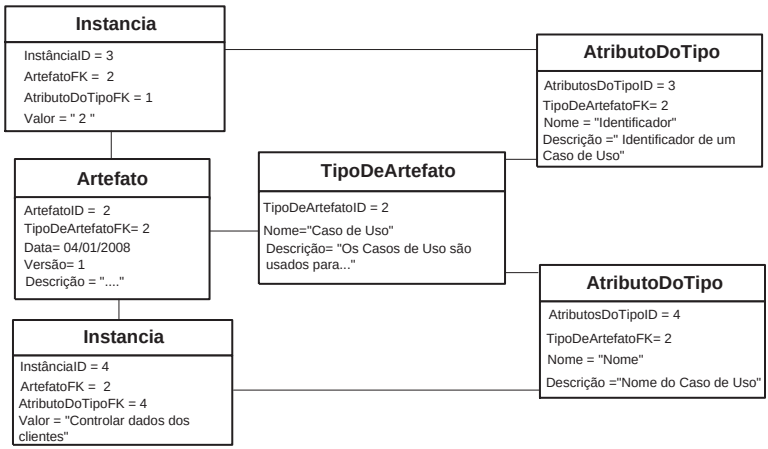


**Figura 6.** Valores do artefato $a_2$.

A partir da criação de $a_1$ e $a_2$ um elo pode ser estabelecido entre eles. Para criar um elo o procedimento é similar ao aplicado aos outros artefatos. O elo a ser criado é um artefato, definido como $a_3$, do tipo *tf*="Elo", que consiste de três atributos *at.nome*="Identificador", $at_1.nome$="Código da Origem", $at_2.nome$="Código do Destino" que tem como valores de instâncias $i_1.valor$=1, $i_2.valor$=1, e $i_3.valor$=2. A Figura 7, ilustra $a_1 \rightarrow a_2$ que é executado por $a_3$.

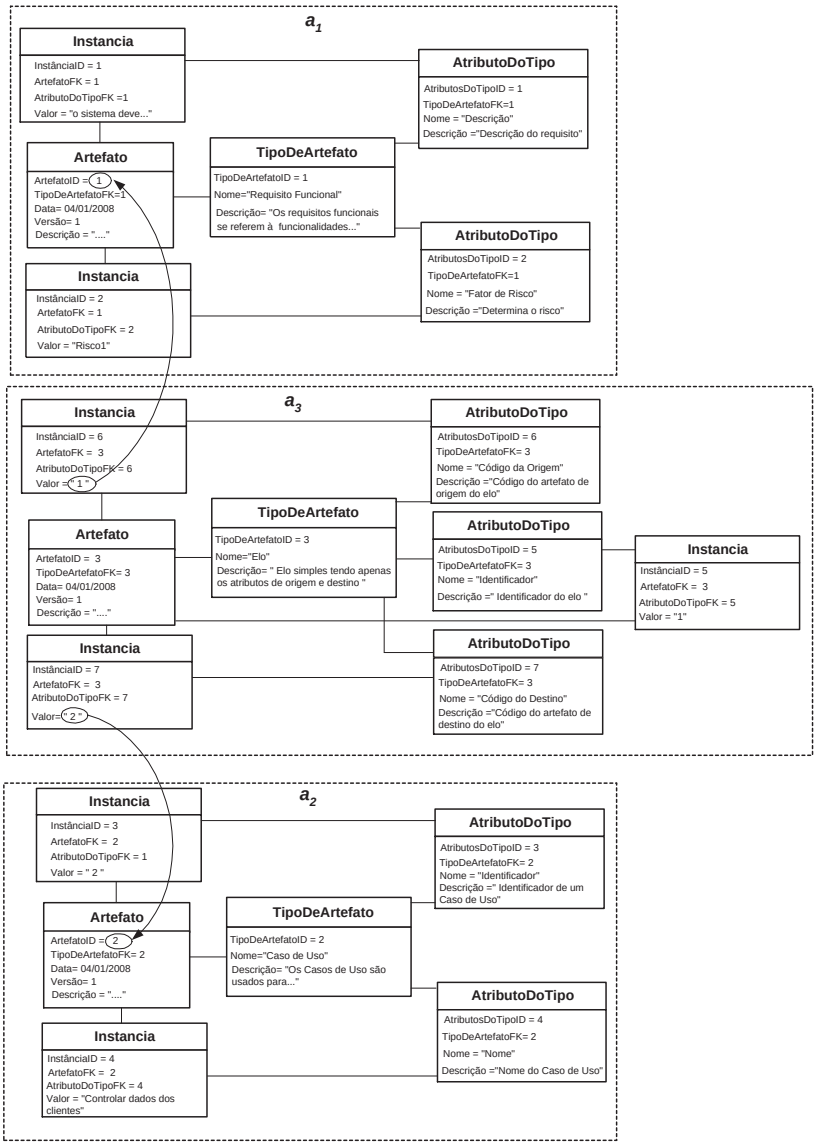


**Figura 7.** Valores do artefato $a_3$. O artefato $a_3$ é do tipo elo usado para relacionar os artefatos $a_1$ e $a_2$.

O elo apresentado por $a_3$ pode ser visualizado usando as operações de projeção($\pi$),

seleção ($\sigma$) e junção natural ($\bowtie$):

$$\pi_{a.ArtefatoID,at.AtributoID,tf.Nome,i.Valor}(\sigma_{a.ArtefatoID=3}(a \bowtie i \bowtie tf \bowtie at). \quad (2)$$

O resultado da Expressão 2 é apresentado na Tabela 4.

**Tabela 4.** Resultado da Expressão 2.

| a.ArtefatoID | at.AtributoID | ak.Nome | i.Valor |
|---|---|---|---|
| 3 | 5 | Identificador | 1 |
| 3 | 6 | Código da Origem | 1 |
| 3 | 7 | código do Destino | 2 |

Através do resultado da Expressão 2, é possível verificar que *i.Valor* apresenta os valores que correspondem ao código do artefato de origem e o código do artefato de destino referentes a um elo. Além dos atributos de identificação (origem e destino) outros atributos podem ser adicionados ao elo, o que permite aumentar a capacidade desse elemento dentro do processo de rastreabilidade. O modelo permite que $n$ novos atributos sejam inseridos em um determinado tipo de elo, desta forma atributos que podem ser utilizados para a elaboração de métricas como tempo, esforço, custo, padrões para *Rationales* ou risco podem ser inseridos ao tipo de elo, tal propriedade amplia a capacidade de elaboração métricas para uso, tanto do processo de rastreabilidade quanto para outros processos do projeto.

## 4 Conclusão

A maior desvantagem dos modelos existentes é que estes estão vinculados a tipos de elos pré–definidos. Nestes modelos quando existe a possibilidade de criação de um novo tipo de elo este se torna um subtipo do elo já pré–definido, o que pode ocasionar a herança de comportamento, quando estas são definidas, o que nem sempre é desejado.

A vantagem em permitir a definição de novos tipos de elos e de novos atributos é a criação de modelos intermediários o que aumenta o domínio dos modelos existentes. Como exemplo, o elo Agregação é definido em alguns modelos enquanto o elo Evolução é definido em outros. O modelo proposto pode incluir ambos e, adicionalmente, permitir a criação de tipos de subtipos através de atributos que determinem um dado artefato como um subelemento.

A principal vantagem do modelo proposto é a generalização sobre o domínio dos tipos de elos de rastreabilidade e a agregação de atributos descritivos para esses elos.

A generalização permite a criação de novos tipos, mas ela aumenta a complexidade para implementar sistemas de rastreabilidade. Isso ocorre devido ao fato de que a criação de

novos atributos, para uma dada necessidade, pode não pertencer ao grupo de restrições do modelo. A definição desses atributos implica um maior controle para que a integridade dos dados não seja afetada.

Uma solução para a criação de novos tipos de elos é mostrar aos usuários, quando iniciar um novo processo de rastreabilidade, um grupo pré-definido de elos que possibilitam a implementação de novos atributos, tanto para os novos elos quanto para os pré–definidos.

Além da criação de tipos de elos, a definição de atributos para um novo tipo pode permitir a redução do número de matrizes de rastreabilidade, pois informações como *Rationales* não precisam ser expressas em uma matriz específica. Desta forma, uma única matriz entre requisitos e componentes pode possuir informações sobre satisfação, *Rationales*, dependência ou custo.

Outro importante ponto que envolve a inclusão de atributos a elos trata sobre definição de métricas para a rastreabilidade. Como observado na seção 2, poucos são os trabalhos que exploram a qualidade da rastreabilidade assim como a definição de métricas para esse processo. O modelo habilita a incorporação de novos atributos para os elos permitindo que sejam adicionadas as métricas já apresentadas ou novos padrões associados a tempo, custo e esforço, tanto para o processo quanto para os artefatos, o que pode facilitar outras atividades do processo de desenvolvimento.

Trabalhos futuros devem explorar alternativas que conduzam a um conjunto de elos que sejam adequados as necessidades da aplicação, o que envolve tanto as necessidades de qualidade estabelecidas por usuários, desenvolvedores, clientes e fornecedores. Entre essas alternativas o uso de mecanismos de inteligência artificial, como sistemas especialistas ou agentes, poderiam ser usados para monitorar e analisar as alterações nos elos e seus resultados aplicados para explorar as possibilidades sobre métricas para a rastreabilidade.

## Referências

[1] SWEBOK – Guide to the software engineering body of knowledge: trial version(version 1.00), IEEE Computer Society Press, May 2004.

[2] Sommerville, I., Sawyer, P., Requirements engineering: a good practice guide, 1nd edition, John Wiley & Sons, July 1998.

[3] Brooks, F.P.Jr., The mythical man–month: essays on software engineering, 2nd edition, Addison Wesley,(Reading, Massachusetts, 1995).

[4] Nuseibeh, B., Easterbrook, S. Requirements engineering: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering (ICSE'00), pp. 35–46, 2000.

[5] Ramesh, B. Jarke, M. Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering*, 27 (1), 58–93, 2001.

[6] Cleland–Huang, J. Chang, C.K., Christensen, M. Event–Based Traceability for Managing Evolutionary Change. IEEE Transactions on Software Engineering, 29 (9),796–810, 2003.

[7] Letelier, P. A Framework for Requirements Traceability in UML–based Projects. In: Proceedings of the International Workshop on Traceability in Emerging Forms of Software Engineering, pp. 30–41, 2002.

[8] Pinheiro, F.A.C., Goguen, J.A. An Object–Oriented Tool for Tracing Requirements. *IEEE Software*, 13 (2), 52–64, 1996.

[9] Toranzo, M., Castro, J., E. Mello. Uma proposta para melhorar o rastreamento de requisitos. In: Proceedings of the Workshop de Engenharia de Requisitos, pp. 194–209, 2002.

[10] Dick, J. Rich Traceability. In: Proceedings of the International Workshop on Traceability in Emerging Forms of Software Engineering, at the 17th IEEE Conference on Automated Software Engineering (ASE), 2002.

[11] Dick, J. Design Traceability. *IEEE Software* November/December, 14–16, 2005.

[12] Scheer, A. W. Business Process Engineering: Reference Models for Industrial Enterprises. Springer-Verlag, 1998.

[13] Palmer, J.D. Traceability. (in R.H. Thayer and M. Dorfman (eds.).*Software Requirements Engineering*. 2nd, IEEE Computer Society Press, p. 412–422), 2000.

[14] Davis, A. M. *Software requirements: objects, functions and states*, Prentice–Hall, New Jersey, 1993.

[15] Dömges, R. Pohl, K. Adapting Traceability Environments to Project–Specific Needs. *Communications of the ACM*, 41(12), 54–62, 1998.

[16] Pinheiro, F. A. C. *Requirements Traceability*, Chapter of the Book Perspectives On Software Requirements. Kluwer Academic Publishers, 2003.

[17] Sayão, M. Leite, J. C. S. P. Rastreabilidade de Requisitos. *Revista de Informática Teórica e Aplicada*, 13 (1), 57–86, 2006.

[18] Spanoudakis, G., Zisman, A., Péres–Miñana, E., Krause, P. Rule–based generation of requirements traceability relations, *The Journal of Systems and Software*, 72, 105–127, 2004.

[19] Software Engineering Institute. CMMI for Development (CMMI–DEV, V1.2). CMU/SEI–2006–TR–008, Software Engineering Institute, Carnegie Mellon University. 2006.

[20] ISO/IEC 15504–2:2003/Cor.1:2004(E). Information Technology – Process Assessment – Part 2: Performing an Assessment. International Organization for Standardization: Geneva, 2003.

[21] Costello, R., J., Liu, D. Metrics for requirements engineering. *Journal of System and Software*, 29(1), 39–63, 1995.

[22] Hull, E., Jakson, K., Dick, J.*Requeriments Engineering*. Spring Verlag, London, September, 2002.

[23] Gotel, O.C.Z. Finkelstein, A. An Analysis of the Requirements Traceability Problem. In: Proceedings of the First Int'l Conf. Requirements Eng., pp. 94–101, 1994.

[24] Egyed, A. Resolving Uncertainties during Trace Analysis. In: Proceedings of the 12th ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE), pp. 3–12, 2004.

[25] Sabetzadeh, M. Easterbrook, S. Traceability in Viewpoint Merging: A Model Management Perspective. In: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering – TEFSE, pp. 44–49, 2005.

[26] Aizenbud–Reshef, N., Nolan, B.T., Rubin, J., Shaham–Gafni, J. Model traceability, *IBM Systems Journal*, 45 (3), 515–526, 2006.

[27] Almeida, J. P., Eck, P. V., Iacob, M. Requirements Traceability and Transformation Conformance in Model–Driven Development. In: Proceedings of the 10th IEEE international Enterprise Distributed Object Computing Conference –Edoc'06, pp. 355–366, 2006.

[28] Jarke, M.Requirements Traceability, *Communications of ACM*, 41 (12), 32–36, 1998.

[29] Papadopoulou, P. Evaluation of a requirements traceability system, MSc Thesis, Department of Computing, City University, 2002.

[30] Riebisch, M., Hubner, M. Traceability–Driven Model Refinement for Test Case Generation. In: Proceedings of the 12th IEEE international Conference and Workshops on the Engineering of Computer–Based Systems –(Ecbs'05), pp. 113–120, 2005.

[31] Pohl, K. PRO–ART: Enabling Requirements Pre–Traceability. In: Proceedings of the 2nd International Conference on Requirement Engineering, pp. 76–84, 1996.

[32] De Lucia, A., Fasano, F., Oliveto, R., Tortora, G. ADAMS Re–Trace: a Traceability Recovery Tool. In: Proceedings of the 9th European Conference on Software Maintenance and Reengineering –CSMR'05, pp. 32–41, 2005.

[33] Evans, M.W. *The Software Factory*, John Wiley and Sons, 1989.

[34] West. M. Quality Function Deployment in Software Development. In: Proceedings of the IEE Colloquium on Tools and Techniques for Maintaining Traceability During Design, pp. 5/1–5/7, 1991.

[35] Jackson, J.A Keyphrase Based Traceability Scheme. In: Proceedings of the IEE Colloquium on Tools and Techniques for Maintaining Traceability During Design, pp. 2/1–2/4, 1991.

[36] Lefering, M. An Incremental Integration Tool Between Requirements Engineering and Programming in the Large. In: Proceedings of the IEEE International Symposium on Requirements Engineering, pp. 82–89, 1993.

[37] Alexander, I. Toward Automatic Traceability in Industrial Practice. In: Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering, pp. 26–31, 2002.

[38] Glinz, M. A Lightweight Approach to Consistency of Scenarios and Class Models. In: Proceedings of the Fourth Int'l Conf. Requirements Eng., 2000.

[39] Kaindle, H. The Missing Link in Requirements Engineering. *ACM SIGSOFT Software Engineering Notes*, 18 (2), pp.30–39, 1993.

[40] Cooke, J., Stone, R.A Formal Development Framework and Its Use to Manage Software Production, Tools and Techniques for Maintaining Traceability During Design. In: Proceedings of the IEEE Colloquium Computing and Control Division, pp. 10/1,1991.

[41] Antoniol, G. Casazza, G. Cimitile, A. Traceability Recovery by Modeling Programmer Behavior. In: Proceedings of the Seventh IEEE Working Conf. Reverse Eng., pp. 240–247, 2000.

[42] Tryggeseth, E., Nytrø, O. Dynamic Traceability Links Supported by a System Architecture Description. In: Proceedings of the IEEE International Conference on Software Maintenance, pp. 180–187, 1997.

[43] Tang, M. X. A knowledge–based architecture for intelligent design support. *The Knowledge Engineering Review*. 12(4), 387–406, 1997.

[44] Smithers, T., Tang, M.X., Tomes, N. The Maintenance of Design History in AI–Based Design. In: Proceedings of the IEE Colloguium on Tools and Techniques for Maintaining Traceability During Design, pp. 8/1, 1991.

[45] Bowen, J., O'Grady, P., Smith, L. A Constraint Programming Language for Life–Cycle Engineering. *Artificial Intelligence in Engineering*, 5 (4), 206–220, 1990.

[46] Faisal, M.H. Toward automating the discovery of traceability links. Doctoral Theses University Of Colorado, p. 118, 2005.

[47] Chechik, M., Gannon, J. Automatic Analysis of consistency between Requirements and Designs. *IEEE Transactions on Software Engineering*, 27 (7),651–672, 2001.

[48] Jung, J. J. Ontological framework based on contextual mediation for collaborative information retrieval. *Information Retrieval*, 10 (1), 85–109, 2007.

[49] Lancaster, F. W. *Information Retrieval Systems: Characteristics, Testing and Evaluation*, Wiley, New York, 1968.

[50] van Rijsbergen, C. J.*Information Retrieval*, 2nd edition, University of Glasgow, London, Butterworths, 1979.

[51] De Lucia, A., Fasano, F. Oliveto, R., Tortora, G. Enhancing an Artifact Management System with Traceability Recovery Features. In: Proceedings of the 20th International Conference on Software Maintenance, pp. 306–315, 2004.

[52] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41, 391–407, 1990.

[53] Deng, M., Stirewalt, R.E., Cheng, B.H. Retrieval By Construction: A Traceability Technique to Support Verification and Validation of UML Formalizations. *International Journal of Software Engineering and Knowledge Engineering*–(IJSEKE), 15 (5), 837–872, 2005.

[54] Hayes, J. H., Dekhtyar, A., Osborne, E. Improving Requirements Tracing via Information Retrieval. In: Proceedings of the IEEE Requirements Engineering Conference, pp. 138–150, 2003.

[55] Marcus, A., Maletic, J. Recovering Documentation–to–Source–Code Traceability Links using Latent Semantic Indexing. In: Proceedings of the IEEE International Conference on Software Engineering, pp. 125–132, 2003.

[56] Spanoudakis, G. Plausible and Adaptive Requirements Traceability Structures. In: Proceedings of the 14th ACM International Conference on Software Engineering and Knowledge Engineering, pp. 135–142, 2002.

[57] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E. Recovering Traceability Links between Code and Documentation. *IEEE Transactions on Software Engineering*, 28 (10), 970–983, 2002.

[58] Zisman, A., Spanoudakis, G., Péres-Miñana, E., Krause, P. Tracing software engineering artifacts. In: Proceedings of the 2003 International Conference on Software Engineering Research and Practice (SERP'03), pp. 448–455, 2003.

[59] Murphy, G. C., Notkin, D. Sullivan, K. J. Software reflexion models: Bridging the gap between design and implementation. *IEEE Transactions on Software Engineering*, 27(4), 364–380, 2001.

[60] Richardson, J., Green, J., 2004. Automating traceability for generated software artifacts. In: Proceedings of the 19th IEEE International Conference on Automated Software Engineering (ASE'04), pp. 24–33.

[61] Cleland-Huang, J., Chang, C.K., Hu, H., Javvaji, K., Sethi, G., Xia, J., 2002. Requirements Driven Impact Analysis of System Performance. In: Proceedings of the IEEE Joint Conference on Requeriments Engineering, pp. 289–296.

[62] Cleland–Huang, J., Settimi, R., Lukasik, W., Chen, Y. Dynamic Retrieval of Impacted Software Artifacts. In: Proceedings of the Midwest Software Engineering Conference, 2003.

[63] Gross, D., Yu, E. From Non–Functional Requirements to Design through Patterns. *Requirements Engineering Journal*, 6 (1), 18–36, 2001.

[64] Cleland–Huang, J., Schmelzer, D. Dynamically Tracing Non–Functional Requirements through Design Pattern Invariants. In: Proceedings of the Workshop on Traceability in Emerging Forms of Software Engineering, 2003.

[65] Mellor, S. J., Clark, A. N., Futagami, T. Model–Driven Development.*IEEE SOFTWARE*, 20(5), 14–18, 2003.

[66] Genvigir, E. C.; Sant'anna, N.; Borrego, F. Modelagem de processos de software através do SPEM – Software Process Engineering Metamodel Specification – Conceitos e Aplicação. In: Workshop dos Cursos de Computação Aplicada do INPE – Instituto Nacional de Pesquisas Espaciais, pp. 85–90, 2003.

[67] Object Management Group, Software Process Engineering Metamodel Specification (SPEM),Formal Submission, OMG document number formal/05–01–06, January 2005.