Uma Infra-Estrutura para Integração de Ferramentas CASE

Rodrigo Oliveira Spinola[1], Marcos Kalinowski[1], Guilherme Horta Travassos[1]
[1]Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ,
Caixa Postal 68511, 21941-972, Rio de Janeiro RJ, Brasil
e-mail: [ros, mkali, ght]@cos.ufrj.br

Resumo

Abordagens para integração de ferramentas CASE têm sido desenvolvidas, em particular, na área de ambientes de desenvolvimento de software (ADS). Normalmente, estas abordagens são baseadas no modelo canônico do ADS ou permitem a integração de ferramentas aos pares. Em conjunto com a evolução tecnológica descrita na literatura, configurando novos desafios para a integração de ferramentas, um conjunto de requisitos para a elaboração de uma nova abordagem de integração visando lidar com as heterogeneidades semânticas e estruturais foi definido. Esta abordagem foi implementada na ferramenta xMapper e sua viabilidade avaliada através de um estudo de caso. Neste artigo, descrevemos a utilização da xMapper, os resultados do estudo de caso e sua aplicação para integrar ferramentas externas a uma infra-estrutura de apoio ao processo de inspeção de software.

Abstract

CASE Tools integration approaches have been developed into the software development environments (SDE) context. Usually, these approaches are based on the SDE canonical model or peer-application needs. Together with the technological evolution described by the technical literature, we hade identified a requirements set to build an innovative integration approach that deals with semantical and syntactical heterogeneity. Such approach has been implemented by a CASE tool named xMapper. A case study to evaluate its feasibility was also performed. This paper describes xMapper, the case study results and shows xMapper using to integrate real CASE tools into a software inspection process supporting infra-structure.

1. Introdução

A tarefa de integração é por natureza complexa e árdua [8]. Representa um campo de pesquisa presente em diversas áreas da computação [6] e muito trabalho tem sido executado no desenvolvimento de tecnologias que permitam a captura e integração da semântica e estrutura distribuídas nos sistemas que se deseja integrar.

Na engenharia de software, abordagens de integração têm sido desenvolvidas, em particular, na área de ambientes de desenvolvimento de software. Existem duas abordagens principais de integração: a baseada em um modelo canônico e a fundamentada na integração de aplicações aos pares. Na perspectiva da utilização de um modelo canônico, todas as ferramentas devem estar projetadas para lidar com uma mesma tecnologia de armazenamento e ainda possuir interfaces para permitir a integração em níveis de apresentação, controle, processo, dado e conhecimento [17]. Caso exista necessidade de integrar alguma ferramenta que não tenha sido projetada para lidar com o modelo canônico, deve ser feita uma adaptação.

Já a abordagem que considera a integração entre pares de aplicações é menos acoplada e, conceitualmente, foi projetada para facilitar o intercâmbio de informação entre aplicações independentes.
Entretanto, corre-se o risco da manutenção do mapeamento entre as ferramentas exigir muito esforço, já que há a necessidade de criação de dois conversores para transformação dos artefatos para cada par de aplicações integradas.
Entre as dificuldades de integração, se destacam aquelas relacionadas a lidar com as heterogeneidades semânticas e estruturais. Além destas duas, pode se descrever também as seguintes [5][11]:

- **Ferramentas Externas:** podem existir mais de uma ferramenta com o mesmo propósito sendo utilizadas em uma mesma atividade. Ou até mesmo, ferramentas contemplando atividades diferentes e complementares durante o desenvolvimento;
- **Abstração de Dados:** abstrair dados é poder manipulá-los independente da aplicação que os criou. Neste caso, o significado da palavra independente deve ser amenizado a depender do objetivo do mecanismo de integração em questão;
- **Evolução Semântica:** o domínio de um artefato pode evoluir em decorrência de novos conceitos. Esta questão influencia diretamente na tecnologia adotada para a construção de um mecanismo de integração. É interessante notar também que uma mudança na semântica pode incorrer na necessidade de redefinição de como os artefatos serão integrados;
- **Integridade Semântica:** a semântica de um domínio e esquemas definem restrições para que o artefato seja interpretado corretamente. Desta forma, é necessário assegurar que a semântica e o esquema sejam mantidos durante a integração;
- **Contexto da Informação:** sempre que uma informação puder ser interpretada por mais de uma maneira, o que nos informará seu verdadeiro significado é o domínio do assunto sendo discutido. Por exemplo, a palavra manga por si só não traz consigo seu significado; ela pode significar uma fruta como também parte de uma roupa, e;
- **Distribuição:** está relacionado com o acesso das ferramentas ao mecanismo de integração.

Neste contexto, a utilização de XML (*eXtensible Markup Language*) tem sido vista como uma solução para as dificuldades de integração. Entretanto, XML possui alguns problemas. Reconhecendo suas limitações, várias comunidades, incluindo a W3C, têm trabalhado em tecnologias correlatas [13]. Alguns desafios no contexto da utilização da XML são [11]:

- **Múltiplos Padrões:** a existência de variados conjuntos de marcações definidos para atender a um mesmo domínio de aplicação dificulta a troca de dados. Para lidar com isto, é possível o desenvolvimento de regras de transformação definidas em XSLT (linguagem para transformação de documentos XML [9]) para efetuar o mapeamento entre as marcações. Entretanto, a especificação e manutenção destas regras de mapeamento não é uma tarefa trivial.
- **Semântica das Marcações:** pensemos agora em uma situação onde há um conjunto de marcações definidas para estruturar o conteúdo de um documento. Ainda assim, o significado que cada marcação possui para cada aplicação é variável. Por exemplo: imaginemos um documento descrevendo a configuração de um computador. A capacidade de armazenamento poderia estar definida pela marcação <*espaçoDisco*>. Entretanto, qual seria a unidade de medida , *megabytes* ou *gigabytes*?

As dificuldades apresentadas até então podem ser classificadas em quatro grupos que têm sua origem:

- Nos possíveis sistemas operacionais, protocolos de rede e *hardware* utilizados por cada ferramenta, **discrepância sistêmica**. Neste grupo se encontra o item distribuição;
- Nas possíveis tecnologias para armazenamento e manipulação dos dados, **discrepância sintática**. Neste grupo se encontram os itens ferramentas externas e abstração de dados;
- Nas diversas formas possíveis de se organizar um documento, **discrepância estrutural**. Neste grupo se encontram os itens ferramentas externas, abstração de dados, integridade semântica, múltiplos padrões e semântica das marcações;
- Nas variadas possibilidades para definição de um mesmo conceito, **discrepância semântica**. Neste grupo se encontram os itens ferramentas externas, abstração de dados, contexto da informação, evolução semântica, integridade semântica, múltiplos padrões e semântica das marcações.

Baseado neste cenário, este artigo apresenta uma abordagem para integração de ferramentas CASE cujos artefatos sejam de um mesmo domínio e descritos em XML. Para isto, ela considera as informações semânticas e estruturais para possibilitar o mapeamento entre artefatos. A partir deste mapeamento, são gerados conversores responsáveis por efetuar a integração entre as ferramentas através de transformações nos artefatos. Essa abordagem lida com os problemas enfrentados pela utilização de um modelo canônico, provendo uma solução menos acoplada, ao mesmo tempo em que reduz o esforço de manutenção exigido para a integração de ferramentas aos pares.

Para permitir sua utilização foi implementada uma ferramenta, tendo sido executado um estudo de caso para avaliar sua viabilidade de utilização na integração de ferramentas CASE reais. Mais especificamente, esta abordagem foi utilizada para tornar possível a instanciação de uma infra-estrutura de apoio ao processo de inspeção de software, onde foram integradas ferramentas independentes para detecção de defeitos em artefatos específicos.

Além desta introdução, este artigo consta de mais quatro seções. Na seção 2, são descritos os requisitos identificados para uma abordagem de integração de ferramentas. A seção 3 apresenta a abordagem para integração proposta. Na seção 4, é descrita a infra-estrutura de apoio ao processo de inspeção e o estudo de caso realizado. Por fim, a seção 5 relata as contribuições deste trabalho frente aos problemas levantados no decorrer do texto.

2. Requisitos para uma Infra-estrutura de Apoio à Integração de Ferramentas

Este tópico apresenta os requisitos identificados para a elaboração de uma abordagem de integração que lide com os quatro grupos de discrepância definidos na introdução. Vale ressaltar que os requisitos para lidar com os problemas mais genéricos como os citados nos itens ferramentas externas e abstração de dados correspondem ao conjunto de requisitos definidos para cada uma das subseções seguintes.

2.1. Discrepância Sistêmica

Contemplar em uma solução de integração mecanismos que tornem possível o envio de informações entre as aplicações e entre estas e o integrador é uma característica essencial. Entretanto, a necessidade de incorporar facilidades de comunicação entre aplicações pode variar de acordo com a arquitetura de integração utilizada. Para arquiteturas baseadas em repositórios centrais, prover internamente facilidades para envio de informações entre aplicações pode ser considerado um requisito básico. Para arquiteturas menos acopladas, a solução de comunicação pode ser feita de forma mais independente. Já a comunicação entre

as aplicações e o mecanismo de integração deve estar presente em ambos os tipos de arquiteturas.

2.2. Discrepância Sintática

Neste caso, é necessário definir uma sintaxe comum para a representação dos artefatos das ferramentas a serem integradas. Assim, deve-se selecionar uma tecnologia para estruturação de informações que esteja sendo amplamente utilizada para evitar a necessidade de criação de uma outra camada para integração com sistemas legados.

2.3. Discrepância Estrutural e Semântica

Estes são dois pontos chave do problema da integração e estão discutidos em conjunto por serem fortemente relacionados, já que a estrutura de um documento ajuda na descrição de sua semântica. Lidar com as disparidades semânticas e estruturais entre artefatos implica em criar mecanismos que tornem possível o mapeamento dos dados. Por ser uma tarefa complexa e muitas vezes repetitiva, o ideal é que um mecanismo de integração disponibilize apoio semi-automatizado a esta atividade. Este apoio implica em outras necessidades como notações para especificar e representar a estrutura e semântica dos artefatos, capturada através de ontologias. Frente às dificuldades apresentadas na introdução deste artigo, uma solução de integração para lidar com:

- **contexto da informação** deve limitar o escopo dos artefatos a serem integrados para minimizar problemas como ambigüidade;
- **a evolução semântica** deve possibilitar que a semântica definida para um artefato possa evoluir. Ou seja, a notação para representação da semântica deve ser extensível. Além disso, deve minimizar o esforço de mapeamento e a geração de conversores caso a semântica do artefato seja alterada;
- **a integridade semântica** deve limitar e guiar a interação do usuário durante o mapeamento entre as fontes de informação para minimizar o problema de definições incorretas durante o mapeamento.

Por fim, outro requisito que deve ser considerado neste item é a forma como os conversores são gerados. Para facilitar, deve ocorrer de forma automatizada com base em informações estruturais e semânticas previamente mapeadas.

3. Abordagem Proposta para Integração de Ferramentas

A Figura 1 apresenta a metáfora da infra-estrutura de baixo acoplamento para integração de ferramentas CASE cujos artefatos sejam de um mesmo domínio e descritos em XML. Esta infra-estrutura permite que usuários possam utilizar softwares que lhes sejam mais familiares na execução de suas atividades e compartilhar informações com outros usuários envolvidos em atividades semelhantes utilizando ferramentas diferentes.

Para realizar a integração, esta metáfora utiliza o seguinte princípio de funcionamento. Um mapeamento é efetuado entre os artefatos das ferramentas que se deseja integrar e a partir dele um conversor, que permite a transformação entre esses artefatos, é gerado (ver Figura 2).

Com base nesta metáfora e nos problemas relacionados às discrepâncias sistêmicas, sintáticas, estruturais e semânticas, foi elaborada a arquitetura apresentada na Figura 3.

Algumas das principais características da infra-estrutura proposta são: (1) utilização da XML como padrão para intercâmbio de informações; (2) uso das tecnologias correlatas à XML; (3)

separação entre o mecanismo de integração e transporte dos artefatos; (4) consideração da semântica das informações a serem integradas através do uso de ontologias; (5) mapeamento semi-automatizado das informações estruturais dos esquemas nas ontologias e; (6) geração automatizada dos conversores.

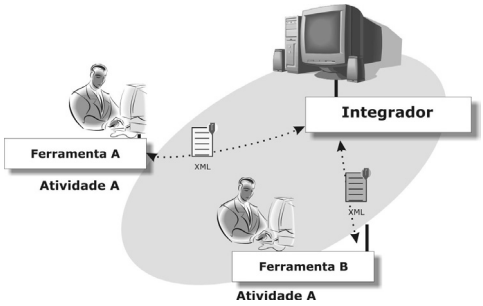

Figura 1. Metáfora da infra-estrutura para Integração de Ferramentas.

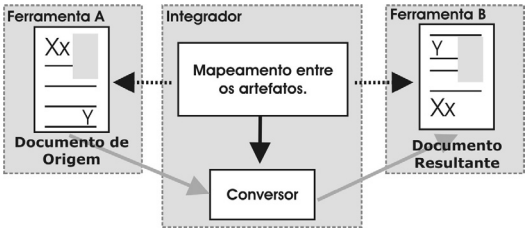

Figura 2. Princípio de Funcionamento.

Alguns fatos que apoiaram a decisão de utilizar XML e suas tecnologias correlatas foram: (1) a disponibilidade de uma grande quantidade de ferramentas que auxiliam a manipulação de arquivos XML [2]; (2) a presença crescente de interfaces para importação e exportação de dados descritos em XML nas aplicações [2]; (3) o fato desta tecnologia ser o padrão para intercâmbio de dados na Internet [1][6]; (4) o suporte do W3C que tem trabalhado em tecnologias correlatas como XSLT, XML *Schema* [3] e *Web Ontology Language* (OWL) [4] e; (5) o fato da utilização da linguagem XML servir de base sintática para o intercâmbio de informações entre as aplicações.

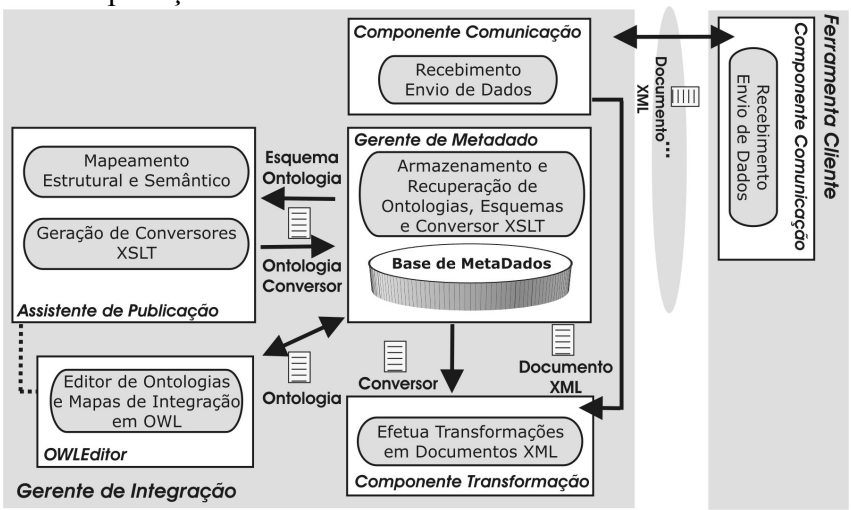

Figura 3. Arquitetura para Integração de Ferramentas CASE.

Este último item é de grande importância pois lida parcialmente com a discrepância sintática, pelo fato de poderem existir sistemas legados que não exportam seus dados em XML. Neste

caso, torna-se necessário a criação de um conversor para transformar os dados legados para o formato XML. A integração com sistemas legados não é contemplada nesta proposta uma vez que o escopo definido é o de artefatos de um mesmo domínio descritos em XML.
Para possibilitar o intercâmbio de informação, a infra-estrutura é composta de duas ferramentas principais: Cliente e Gerente de Integração. A ferramenta cliente é responsável, através do componente comunicação, em permitir ao usuário ter acesso, de forma transparente, às funcionalidades do Gerente de Integração.
O Gerente de Integração é o foco da infra-estrutura de integração e é nele que são tratadas as questões referentes aos mapeamentos estruturais e semânticos dos artefatos para que seja possível a geração de conversores. O Gerente de Integração mapeia os esquemas dos documentos a serem integrados em um mapa de integração. Este é um documento criado a partir de uma ontologia, descrevendo a semântica de um artefato, acrescida de informações estruturais provenientes dos esquemas dos artefatos. Ele centraliza informações semânticas e estruturais e é a base para a geração de conversores. Assim, é possível fazer uso da semântica descrita em uma ontologia para a definição de vários mapas de integração entre ferramentas que se deseja integrar (ver Figura 4).

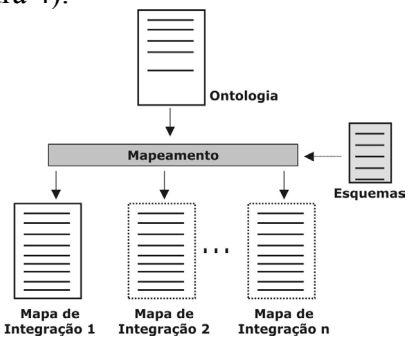

Figura 4. Criação de Mapas de Integração.

A partir de agora serão apresentados, em detalhes, os componentes que formam a infra-estrutura e como lidam com os problemas envolvendo as heterogeneidades sistêmica, semântica e estrutural.

3.1. Gerente de Integração

O Gerente de Integração possui as seguintes responsabilidades: (1) efetuar transformações nos documentos XML; (2) gerenciar metadados (esquemas, conversores, ontologias e mapas de integração); (3) estabelecer mecanismos para mapeamento semi-automatizado entre informações estruturais e semânticas; (4) criar de forma automatizada conversores (XSLTs); (5) possibilitar a definição da semântica dos artefatos a serem integrados através de ontologias e; (6) tornar transparente o acesso ao mecanismo de integração para seus usuários. É nele que são contempladas as facilidades responsáveis por tratar os problemas de integração descritos na introdução. Para isto, é formado pelos cinco componentes (ver Figura 3) descritos nas subseções seguintes.

3.1.1. Gerente de Metadado. Todas as ações efetuadas na utilização do mecanismo de integração têm como resultado ou pré-requisito um documento XML seja ele especificando um esquema (XML Schema), uma ontologia (OWL), um mapa de integração ou um conversor (XSLT). Para lidar com estes documentos tornando possível seu armazenamento e recuperação, foi criado o componente Gerente de Metadado. Suas principais funcionalidades

são: (1) organizar a estrutura de armazenamento; (2) salvar os variados tipos de documentos manipulados pelo Gerente de Integração; (3) tornar possível a remoção destes documentos e; (4) disponibilizar interfaces para interação com os demais componentes do Gerente de Integração.

3.1.2. OWLEditor. Ontologias podem auxiliar na formalização da semântica presente em artefatos. Entretanto, construí-las não é uma tarefa trivial. Envolve a definição de conceitos, propriedades, axiomas e restrições. Isto implica na necessidade de um apoio ferramental para seu desenvolvimento [10]. Na integração de dados, ontologias podem ser úteis no compartilhamento da informação através da definição de um vocabulário comum. O componente OWLEditor serve para definir ontologias em OWL apoiando duas das atividades presentes em processos para construção de ontologias: captura e formalização. Para isto, ele possui três módulos:

- **Definição de Metadados:** onde são manipuladas informações que descrevem a ontologia.
- **Definição dos Conceitos:** onde são manipulados os conceitos e, as restrições e axiomas relacionados a ele. Este módulo possui também algumas funcionalidades que impedem que um engenheiro do domínio defina relações inconsistentes ou incoerentes na ontologia. Por exemplo, o módulo não permite que um conceito seja ao mesmo tempo equivalente e disjunto a um outro conceito.
- **Definição das Propriedades:** onde são manipuladas as propriedades e as restrições relacionadas a elas. Este módulo também possui uma funcionalidade que impede a definição inconsistente ou incoerente das propriedades. Neste caso, não é possível definir uma propriedade como sendo equivalente e inversa a uma mesma propriedade.

Vale destacar aqui também que este componente auxilia a manutenção dos mapas de integração. Como estes são gerados a partir de ontologias, a manipulação dos conceitos e propriedades presentes no mapa de integração se torna possível.

3.1.3. Assistente de Publicação. Este é o componente chave da infra-estrutura de integração. Nele são efetuados os mapeamentos estruturais e semânticos e, gerado o conversor entre os artefatos. Para criar este mecanismo foram utilizadas as tecnologias XML *Schema*, OWL e XSLT. As duas primeiras voltadas para o mapeamento, a terceira para transformações.

Para lidar com as discrepâncias estruturais e semânticas, poder-se-ia pensar em uma solução onde para cada relacionamento entre aplicações houvesse a necessidade de mapeamento entre seus artefatos. Isto teria como conseqüência a necessidade de desenvolvimento manual de dois conversores para importação e exportação dos dados para cada par de aplicações. Entretanto, a manutenção de uma solução seguindo este princípio seria ineficaz e ineficiente. Para isto, basta atentarmos para o fato de que a quantidade de mapeamentos e conversores mantidos crescem a uma razão quadrática. Percebe-se que a quantidade de mapeamento a serem feitos obedece à expressão $n^2 - n$ onde n é o número de aplicações a serem integradas.

O mecanismo aqui proposto para lidar com este problema é fundamentado na utilização de ontologias para representar a semântica dos artefatos e esquemas contendo as informações estruturais. De forma simplificada, a idéia é mapear os esquemas dos documentos, a serem integrados, na ontologia. A partir do momento que este mapeamento é realizado, a ontologia passará a ser denominada Mapa de Integração. Este conterá informações semânticas e estruturais que guiarão a geração automatizada dos conversores entre as aplicações cujo esquema tenham sido mapeados. Esta solução lida diretamente com o problema citado no parágrafo anterior: (1) a quantidade de mapeamentos a serem feitos passa a crescer de forma

linear uma vez que as informações necessárias para a criação dos conversores só precisam ser mapeadas uma vez para cada aplicação e, (2) os conversores são gerados de forma automatizada diminuindo consideravelmente o trabalho para a integração das ferramentas.
Serão apresentados agora os mecanismos de mapeamento entre artefatos e de geração de conversores.
Mapeando Esquemas no Mapa de Integração. O mapeamento tem como entrada um Mapa de Integração inicial definindo a semântica do artefato e o esquema especificando a estruturação do artefato. O resultado do processamento será a inclusão de um conjunto de informações no Mapa de Integração que permitirá a geração dos conversores. Para definir como este mapeamento é realizado, foi necessário definir algumas diretrizes que serviram de base para a elaboração do algoritmo de mapeamento das informações estruturais no mapa de integração:

- **Diretriz 01:** elementos de tipo complexo (elementos que podem conter atributos e outros elementos) devem ser mapeados para conceitos no mapa de integração;
- **Diretriz 02:** elementos de tipo nativo (aqueles previamente definidos como inteiro, data, *string*...) devem ser mapeados para propriedades no mapa de integração;
- **Diretriz 03:** um atributo de um elemento de tipo complexo ou nativo deve ser mapeado como atributo para seu conceito ou propriedade correspondente e, seu conteúdo pode ter origem em um outro atributo ou conceito do documento de origem;
- **Diretriz 04:** uma seqüência de elementos definidos pela marcação *sequence* no esquema deve ser mapeada no mapa de integração como uma seqüência de elementos dentro do conceito ao qual o elemento de tipo complexo corresponde;
- **Diretriz 05:** a quantidade de vezes que um elemento poderá aparecer definido pela marcação *maxOccurs* ou *cardinality* no esquema deve ser mapeada para o elemento correspondente no mapa de integração, seja ele conceito ou propriedade;
- **Diretriz 06:** gerar um identificador a ser inserido nos elementos mapeados no mapa de integração a partir de seu esquema de forma que seja possível identificar de qual esquema são provenientes as informações mapeadas.

Como a OWL não especifica marcações próprias para efetuar estes mapeamentos, foi criado um conjunto de marcações. Estas marcações guiarão a geração dos conversores XSLT. Como estas marcações serão inseridas em conceitos ou propriedades do mapa de integração, teremos dois sub-conjuntos de marcadores. As Figuras 5 e 6 apresentam as marcações definidas para conceitos e propriedades respectivamente.
Analisando a Figura 5 tem-se o seguinte:

- **Linha 1:** é informado que existe uma informação de esquema mapeada no mapa de integração e de qual esquema as informações são provenientes (*map:id*);
- **Linha 2:** empacota as informações provenientes do esquema para organizar as marcações de mapeamento;
- **Linha 3:** informa quais atributos pertencem àquele conceito no esquema mapeado. Neste caso, pode existir mais de um atributo e, por conseqüência, mais de um marcador *map:attribute*. Este marcador pode possuir também um atributo *element* que indica onde será buscado o conteúdo para o atributo em questão;
- **Linha 4:** informa o nome que aquele conceito representa no artefato do esquema mapeado;
- **Linha 5:** indica que aquele conceito possui propriedades ou outros conceitos e que estes estão organizadas em uma determinada seqüência;

- **Linhas 6 e 7:** apresentam os elementos (propriedades ou outros conceitos) que compõem o conceito atual. Cada marcador *map:next* informa qual o próximo elemento – conceito ou propriedade – a ser mapeado e o atributo *moreOne* indica se aquele elemento mapeado do esquema pode (1) ou não (0) se repetir.

```
1:  <map:mapFile map:id="">
2:          <map:schema>
3:              <map:attribute element=""></map:attribute>
4:              <map:name></map:name>
5:              <map:sequence>
6:                  <map:next moreOne="0"></map:next>
7:                  <map:next moreOne="1"></map:next>
8:              </map:sequence>
9:          </map:schema>
10: </map:mapFile>
```

Figura 5. Marcações para mapeamento nos conceitos do mapa de integração.

Estes marcadores contemplam as diretrizes 01, 03, 04, 05 e 06. Agora, analisando a Figura 6, tem-se o seguinte:

- **Linha 1:** é informado que existe uma informação de esquema mapeada no mapa de integração e de qual esquema as informações são provenientes (*map:id*);
- **Linha 2:** empacota as informações provenientes do esquema para organizar as marcações de mapeamento;
- **Linha 3:** informa o nome que aquela propriedade representa no artefato do esquema mapeado.

```
1: <map:mapFile map:id="">
2:      <map:schema>
3:              <map:name></map:name>
4:      </map:schema>
5: </map:mapFile>
```

Figura 6. Marcações para mapeamento nas propriedades do mapa de integração.

Este conjunto de marcadores dá suporte às diretrizes 02, 03 e 06.

Tendo definido as diretrizes e o conjunto de marcações necessárias, foi então desenvolvido um algoritmo que efetuasse o mapeamento estrutural/semântico. O algoritmo está fundamentado na comparação entre os nomes dos conceitos e propriedades do esquema e, do mapa de integração para efetuar o mapeamento. Encontrando similaridades entre o nome de um elemento do esquema e um da ontologia, as diretrizes 01 ou 02 são aplicadas e com isso, as marcações de mapeamento são inseridas no mapa de integração.

Como o mapeamento é baseado em similaridades entre os nomes dos elementos, propriedades e conceitos, existe a possibilidade do algoritmo não efetuar o mapeamento de alguns elementos. Por este motivo, a abordagem de mapeamento aqui proposta é semi-automatizada e os nomes dos conceitos e das propriedades não mapeadas são armazenadas em uma lista para que o engenheiro informe como será realizado seu mapeamento.

Gerando Conversores XSLT. Este módulo do Assistente de Publicação é o responsável pela geração automatizada dos conversores descritos em XSLT. Para isto, ele executa um algoritmo que utiliza as informações semânticas e estruturais armazenadas no mapa de integração durante o procedimento descrito na seção anterior. O algoritmo em questão possibilita a escolha de um documento de origem e vários alvos. A geração dos conversores sempre se dá aos pares, documento origem → documento alvo, uma vez que as regras de

transformação definidas em arquivos XSLT deverão buscar alguma informação no documento de origem e estruturá-la de outra maneira no documento alvo.
Embora as marcações mapeadas a partir dos dois esquemas sejam utilizadas, as que servirão de base para definição do documento gerado a partir do conversor serão provenientes do esquema do documento alvo. Isto porque são elas que informarão a estrutura que o artefato alvo deverá possuir. As informações do esquema de origem serão utilizadas apenas para informar quais marcadores do documento de origem se transformarão em seu correspondente no documento alvo. Assim, para cada elemento do esquema do artefato alvo que tenha sido mapeada, será procurado seu correspondente nas informações do esquema de origem. Caso esta informação esteja disponível, é gerada uma regra para transformações entre os dois elementos e uma instrução para recuperar o valor do documento de origem e colocá-la no artefato alvo. Caso a informação não esteja disponível, será criada uma regra que definirá o marcador necessário no artefato alvo e seu conteúdo será vazio já que esta informação não está presente no artefato de origem.
Esta funcionalidade complementa o módulo responsável por efetuar o mapeamento no sentido de lidar com as questões envolvendo discrepâncias estruturais e semânticas. A partir destes dois módulos é possível efetuar a integração de ferramentas distintas que queiram compartilhar artefatos de um mesmo domínio descritos em XML.

3.1.4. Componente Transformação. O componente Transformação é o responsável por efetuar as transformações nos dados utilizando para isso as regras de transformação criadas no Assistente de Publicação. De todos os componentes que fazem parte do Gerente de Integração, este é o mais independente. Isto porque as transformações nos documentos XML são efetuadas via XSLT e para isto, é necessária a utilização de um processador XSLT. Para efetuar as transformações nos documentos, basta informar para o processador onde os documentos XML e XSLT podem ser encontrados e onde será gerado o artefato alvo. O processador se encarregará de efetuar as transformações necessárias.

3.1.5. Componente Comunicação. Foi visto na seção 2 que tornar o acesso das ferramentas ao mecanismo de integração transparente é importante na sua construção. Para lidar com esta questão, o Componente Comunicação torna o acesso das ferramentas ao mecanismo de integração transparente e disponibiliza interfaces das funcionalidades providas pelo Gerente de Integração relacionadas à transformação de documentos XML.

3.2. Ferramenta Cliente

A ferramenta Cliente será responsável por permitir a comunicação do usuário com o Gerente de Integração. Para prover esta funcionalidade, ela possui um componente de comunicação.

3.2.1. Componente Comunicação. Este módulo complementará as funcionalidades do componente Comunicação do Gerente de Integração. Junto a ele, proverá uma camada para separar o mecanismo de integração do transporte de dados tornando transparente a comunicação entre a ferramenta Cliente e o Gerente de Integração.

3.3. Processo de Integração

O processo apresentado nesta seção tem como objetivo apoiar a abordagem de integração discutida neste artigo. O processo apresenta as alternativas possíveis de serem seguidas

durante a integração explicitando os artefatos gerados em cada atividade e quem é o responsável por executá-la. A Figura 7 apresenta o processo a ser seguido para configuração do ambiente de integração. O processo é composto pelas seguintes atividades:

- **Configurar Gerente de Metadado:** o engenheiro de software define o local onde serão mantidas as informações necessárias e geradas pela infra-estrutura;
  Produto: arquivo de configuração do gerente de metadado;
  Responsável: Engenheiro de Software;
- **Definir Ontologia:** o engenheiro do domínio define os conceitos e propriedades referentes ao domínio do artefato das ferramentas a serem integradas;
  Produto: ontologia definida;
  Responsável: Engenheiro do Domínio;

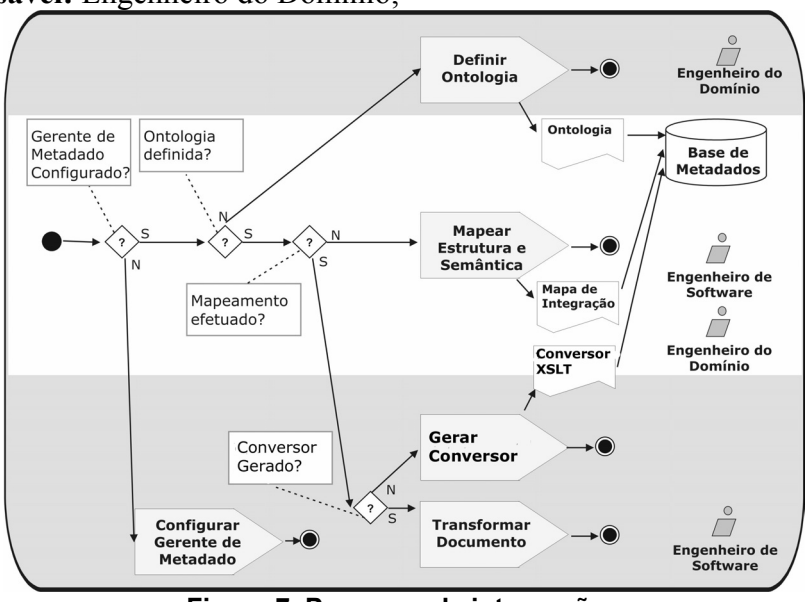

Figura 7. Processo de integração.

- **Mapear Semântica e Estrutura:** o engenheiro do domínio ou de software efetua o mapeamento de artefatos através da inserção de informações estruturais provenientes dos esquemas representando os documentos fonte e alvo no mapa de integração;
  Produto: mapa de integração;
  Responsável: Engenheiro do Domínio ou de Software;
- **Gerar Conversor:** o engenheiro de software gera de forma automatizada o conversor de mapeamento a partir das informações (semânticas e estruturais) contidas no mapa de integração;
  Produto: conversor gerado;
  Responsável: Engenheiro de Software;
- **Transformar Documento:** o engenheiro de software seleciona o artefato a ser transformado e o conversor contendo as regras de transformação.
  Produto: artefato alvo;
  Responsável: Engenheiro de Software;

3.4. xMapper: Apoio Ferramental para a Abordagem de Integração Proposta

Baseado nos requisitos descritos na seção 2 e na abordagem de integração proposta foi implementada a ferramenta xMapper. Ele implementa os componentes Assistente de

Publicação, Gerente de Metadado, OWLEditor e Transformação fornecendo apoio às atividades do processo de integração apresentado na seção 3.3.
A seção seguinte mostra a utilização do xMapper para tornar possível a construção de uma infra-estrutura de apoio ao processo de inspeção de software na qual são integradas ferramentas independentes para detecção de defeitos em artefatos específicos.

4. Aplicação da Abordagem para Integração de Ferramentas CASE na Automatização de Inspeções de Software

O objetivo de inspeções de software é melhorar a qualidade de artefatos de software através de sua análise, detectando e removendo defeitos antes que o artefato seja passado para a próxima fase do processo de desenvolvimento de software. Muito conhecimento tem sido produzido na área de inspeções de software e se mostrado adequado através de estudos experimentais. Entretanto, pouco apoio computacional capturando este conhecimento está disponível. Neste contexto, foi desenvolvido na COPPE/UFRJ uma infra-estrutura de apoio computacional ao processo de inspeção de software [7]. Nas próximas subseções será apresentada a infra-estrutura desenvolvida enfatizando a utilização da abordagem de integração durante a execução de um estudo de caso.

4.1. Infra-Estrutura de Apoio ao Processo de Inspeção de Software

A arquitetura da infra-estrutura para inspeção de software é formada por três tipos de componentes: (1) um arcabouço para a infra-estrutura, (2) ferramentas externas de apoio à aplicação de técnicas de inspeção específicas para determinado tipo de artefato e, (3) um integrador de ferramentas para possibilitar o intercâmbio de dados.
Nesta arquitetura, o arcabouço da infra-estrutura é responsável, dentre outras coisas, por tornar a infra-estrutura acessível pela *web* e garantir que o processo de inspeção seja seguido de forma sistemática. Seu módulo de controle tem conhecimento das ferramentas externas existentes e disponibiliza as apropriadas para o usuário, de acordo com as técnicas que devem ser utilizadas na inspeção que está sendo realizada. Os dados das ferramentas externas são então retornados utilizando o módulo integrador. Uma das características desta arquitetura é o fato dela tornar a infra-estrutura fracamente acoplada através do uso da abordagem de integração, permitindo que ferramentas independentes sejam utilizadas em conjunto para prover mais funcionalidades à infra-estrutura, tornando-a customizável e extensível.
Baseado nesta arquitetura, a infra-estrutura de apoio ao processo de inspeção de software foi implementada. O componente do arcabouço da infra-estrutura recebeu o nome ISPIS [7]. Entre outras características, ISPIS automatiza o controle do processo de inspeção e, na fase de detecção de defeitos disponibiliza ferramentas externas a depender do contexto da inspeção que está sendo realizada. Estas ferramentas de apoio à aplicação de técnicas específicas de detecção de defeitos utilizadas são as atualmente disponíveis na COPPE/UFRJ. Elas apóiam a aplicação de PBR [14] e de OORT's [12]. Os dados gerados pela utilização destas ferramentas podem ser carregados em ISPIS mediante a utilização do conversor gerado pela ferramenta de integração [15]. A configuração atual da infra-estrutura está ilustrada na Figura 8.

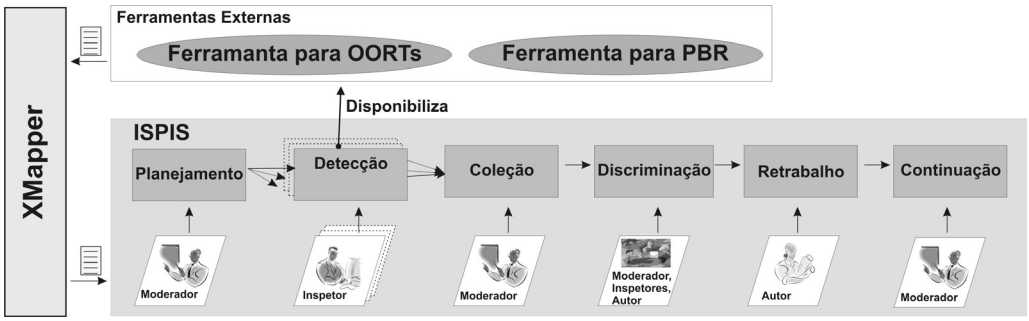

Figura 8. Configuração atual da infra-estrutura de apoio ao processo de inspeção [7].

4.2. Estudo de Caso

Experimentação é o centro do processo científico. No campo da Engenharia de Software, a utilização de métodos experimentais pode trazer alguns benefícios dentre os quais: acelerar o progresso através da rápida eliminação de abordagens não fundamentadas, suposições errôneas e modismos, ajudando a orientar a engenharia e a teoria para direções promissoras [16]. Foi então definido e elaborado um estudo de caso, no contexto da infra-estrutura de apoio ao processo de inspeção de software descrita acima, para analisar a abordagem para integração de artefatos de um mesmo domínio descritos em XML com o propósito de caracterizá-la com respeito à viabilidade de utilizar a abordagem para integração de ferramentas do ponto de vista do desenvolvedor de ferramenta no contexto da utilização da abordagem integrando uma ferramenta de apoio à aplicação de PBR e ISPIS.

Tanto a ferramenta de apoio à aplicação de PBR quanto ISPIS lidam com informações relacionadas à inspeção de software. O artefato considerado neste estudo de caso é o relatório de discrepâncias (uma discrepância é um possível defeito encontrado durante a atividade de detecção de defeitos). Entretanto, ele é estruturado de forma diferente e cada ferramenta possui necessidades específicas de informação. Parte dos artefatos a serem integrados está apresentada nas Figuras 9 e 10. Neste caso, o sentido do intercâmbio da informação é da ferramenta de apoio a PBR para ISPIS.

Para estes artefatos, há a necessidade de omitir algumas marcações (*perspective, project, inspector, artifact*, dentre outras), modificar alguns de seus nomes (*start*, *end*, dentre outras) e, acrescentar a marcação *severity*. Além disto, é necessário mapear a relação de dois ou mais conceitos no artefato de origem para um conceito no artefato destino. Neste caso, o conteúdo proveniente contido nas marcações <*requirement*> e <*line*> do artefato da ferramenta de PBR deveriam ser mapeados para o marcador <*location*> do documento importado por ISPIS.

O estudo de caso utilizou o desenvolvedor de ISPIS para desempenhar o papel do integrador. Sua tarefa foi, utilizando o xMapper, construir o mapa de integração entre as ferramentas. Foi avaliado esforço de integração (medido em tempo) e se houve perda entre os dados integrados de acordo com o mapa de integração produzido. Observou-se que o integrador foi capaz de realizar as atividades do processo de integração com sucesso. O tempo de integração foi de 17 minutos tendo que ser considerado nesse tempo o fato do integrador não ter utilizado anteriormente o xMapper e as dúvidas terem sido esclarecidas durante a integração. Em relação à perda de informações, o artefato foi transformado sem comprometer sua utilização por ISPIS, ou seja, a estrutura e semântica do documento foram mantidas, e também não houve perda de conteúdo.

```
<formDiscrepancy type="pbr">
    <perspective>USER</perspective>
    <project>ESE</project>
    <inspector>103137506</inspector>
    <start>7/9/2003 09:40:56</start>
    <artifact>C:\Program Files\PBRTool\SRS_Documents\ABC_Video_System.rtf</artifact>
    <discrepancyList>
        <discrepancy status="updated">
            <classification>Omissão</classification>
            <description>Como calcular a taxa de aluguel? Que outras taxas existem?</description>
            <identTime>7/9/2003 10:44:44</identTime>
            <step>Participants - View Document</step>
            <line>300</line>
            <requirement>Functional Requirement 8</requirement>
        </discrepancy>
        …
    </discrepancyList>
    <PBR_help>0</PBR_help>
    <Tool_help>0</Tool_help>
    <end>7/9/2003 12:13:47</end>
    <totalTime>02:32:50</totalTime>
</formDiscrepancy>
```

Figura 9. Fragmento do artefato gerado pela ferramenta de apoio à aplicação de PBR.

```
<formDiscrepancy type="pbr">
    <discrepancyList type="pbr">
        <inspectionStart>7/9/2003 09:40:56</inspectionStart>
        <discrepancy>
            <location>Functional Requirement 8 - line: 300</location>
            <classification>Omissão</classification>
            <severity/>
            <description>Como calcular a taxa de aluguel? Que outras taxas existem?</description>
        </discrepancy>
        …
        <inspectionEnd>7/9/2003 12:13:47</inspectionEnd>
    <discrepancyList>
</formDiscrepancy>
```

Figura 10. Fragmento do artefato utilizado por ISPIS.

Feito o estudo, o xMapper passou a ser utilizado como parte da infra-estrutura descrita na seção 4.1. Esta infra-estrutura tem sido utilizada em inspeções de software reais [7].

5. Considerações Finais

Lidar com os problemas de heterogeneidade semântica e estrutural na integração de ferramentas CASE, como discutido neste artigo, implica na necessidade de transformações entre diferentes perspectivas de utilização de um mesmo artefato. O ideal seria que essas transformações fossem efetuadas de forma automatizada, entretanto, criar mecanismos que infiram automaticamente os mapeamentos necessários nem sempre é possível.

Ontologias podem auxiliar este mapeamento, no entanto, elas nem sempre capturam toda a semântica dos artefatos. Um dos motivos disto são as necessidades específicas de cada ferramenta frente a um mesmo tipo de informação. Dessa forma, é necessária a intervenção humana no processo de identificação de correspondências entre diferentes artefatos. Neste caso, o ideal passa a ser: minimizar a intervenção do usuário e quando necessária sua participação, auxiliá-lo no sentido de evitar que a integridade semântica da informação a ser mapeada não seja quebrada.

Neste contexto, esse artigo apresentou uma abordagem para integração de ferramentas CASE, cujos artefatos sejam de um mesmo domínio e descritos em XML, utilizando informações estruturais e semânticas provenientes de esquemas e ontologias, respectivamente. Esta

abordagem lida com os problemas de integração apresentados na introdução deste artigo. Dois desses problemas são básicos para qualquer infra-estrutura que se proponha a lidar com a integração de ferramentas: **abstração de dados** e **ferramentas externas**. Para a questão da abstração de dados, a abordagem limita o escopo de integração a artefatos de um mesmo domínio descritos em XML. Já para lidar com ferramentas externas, basta que estas exportem seus dados no formato XML. Frente às demais dificuldades, a abordagem traz os seguintes benefícios:

- **Distribuição:** parte da arquitetura de integração proposta é responsável por tornar transparente a comunicação entre as ferramentas e o mecanismo de integração.
- **Evolução Semântica:** como foi visto, a principal dificuldade presente neste tópico é conseqüência da própria evolução do domínio do artefato. Caso as modificações afetem as informações já mapeadas, haverá a necessidade de mapear novamente as informações estruturais no mapa de integração e gerar novos conversores. Como a abordagem aqui proposta permite ao usuário realizar isso de forma semi-automatizada e automatizada, respectivamente, o esforço de manutenção é reduzido.
- **Integridade Semântica:** este requisito não é totalmente contemplado. Ele é difícil de ser atingido pelo fato do algoritmo para mapeamento criado ser executado de forma semi-automatizada. Isto dá margem à quebra da integridade nos momentos em que uma decisão deva ser tomada pelo usuário. Embora não lide diretamente com este problema, a solução apresentada tenta minimizá-lo através de controles de integridade durante a definição da semântica dos artefatos e evitando, quando possível, interação com o usuário durante o mapeamento.
- **Contexto da Informação:** o fato dos artefatos considerados para integração serem de um mesmo domínio e terem sua semântica descrita através de ontologias, minimiza ou elimina a possibilidade de ambigüidade ou interpretações incorretas das informações contidas nos documentos.
- **Múltiplos Padrões:** a abordagem proposta minimiza o esforço do integrador de ferramentas na criação e manutenção dos conversores necessários através de um procedimento semi-automatizado.
- **Semântica das Marcações:** esta abordagem não traz nenhum ganho quanto a essa característica.

Esta abordagem foi implementada na ferramenta xMapper. Um estudo de caso permitiu avaliar a viabilidade de sua utilização. Além disto, a ferramenta é atualmente utilizada para prover a integração de ferramentas externas a ISPIS, uma infra-estrutura de apoio ao processo de inspeção de software que está disponível e tem sido utilizada em inspeções reais na COPPE/UFRJ. Neste contexto, o xMapper tem se mostrado uma solução adequada para a integração de ferramentas CASE tanto no processo de construção do mapa de integração para cada uma das ferramentas externas quanto na geração e disponibilização dos conversores. Maiores informações podem ser obtidas em (http://www.cos.ufrj.br/~ese).

6. Agradecimentos

Os autores deste trabalho gostariam de agradecer a CAPES e ao CNPq pelo apoio financeiro e ao projeto NSF-CNPq READERS II e à Equipe de Engenharia de Software Experimental da COPPE/UFRJ (www.cos.ufrj.br/~ese) pelos comentários e sugestões significativas. O prof. Guilherme Travassos reconhece e agradece o apoio fornecido pelo CNPq para o desenvolvimento deste trabalho.

Referências

1. Collins, S.R., Navathe, S., Mark, L., "XML Schema mappings for heterogeneous database access". Information and Software Technologies, v. 44, pp. 251-257, 2002.
2. Emmerich, W., et al., TIGRA – An Architectural Style for Enterprise Application Integration. In Proc. of the 23rd Int. Conference on Software Engineering, Toronto, Canada. pp. 567-576. ACM Press, 2001.
3. Fallside, D.C., XML Schema Part 0: Primer. W3C Recommendation. Disponível em http://www.w3c.org/TR/xmlschema-0/. Último acesso em 13/11/2003.
4. Harmelen, O., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Chneider, P.F., Stein, L.A., OWL Web Ontology Language Reference. Working Draft, World Wide Web Consortium. Disponível em http://www.w3.org/TR/owl-ref/. Acessado em 14/11/2003.
5. Harrison, W., Ossher, H., Tarr, P., *The Future of Software Engineering , Software Engineering Tools and Environments: a Roadmap*. Pp. 261-277, ACM Press, 2000.
6. Hasselbring, W., Information System Integration. Communications of the ACM, vol. 43, nº 6, pp. 33 – 38, 2000.
7. Kalinowski, M., Spinola, R. O., Travassos, G.H., "Infra-Estrutura Computacional para Apoio ao Processo de Inspeção de Software", Simpósio Brasileiro de Qualidade de Software, 2004.
8. Karsai, G., "Design Tool Integration: An exercise in Semantic Interoperability". In 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems April 03 - 07, Edinburgh, Scotland, 2000.
9. Kay, M., XSL Transformations (XSLT) Version 2.0. W3C Working Draft. Disponível em http://www.w3.org/TR/xslt20/. Último acesso em 20/11/2003.
10. Lassila, O., Van Harmelen, F., Horrocksm, I., Hendler, J., McGuinness, D. L., The Semantic Web and its Languages, In: IEEE Intelligent Systems, p. 67-73, November/December, 2000.
11. Madnick, S., "The Misguided Silver Bullet: What XML will and will NOT do to help Information Integration," *Proceedings of the Third International Conference on Information Integration and Web-based Applications and Services*, IIWAS2001, Linz, Austria, September: 1-72, 2001.
12. Reis, L.N.M, Travassos, G.H., "Apoio Automatizado à Configuração e Aplicação de OORT's", Anais do WTES 2003 (SBES) - Workshop de Teses em Engenharia de Software, p.59-64, Manaus.
13. Seligman, L., Rosenthal, A., "XML´s impact on databases and data sharing". Computer, June 2001, pp. 59-67.
14. Silva, L.F.S., Chapetta, W.A., Travassos, G.H., "Inspeções de Requisitos de Software Utilizando PBR e Apoio Ferramental", Simpósio Brasileiro de Qualidade de Software, 2004.
15. Spinola, R.O., Travassos, G.H., "Uma Abordagem para Integração de Ferramentas", Anais do WTES 2003 (SBES) - Workshop de Teses em Engenharia de Software, p.59-64.
16. Travassos, G.H.; Gurov, D.; Amaral, E.A.G.G., Introdução à Engenharia de Software Experimental. In: Relatório Técnico ES-590/02-Abril, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, disponível em http://www.cos.ufrj.br/publicacoes.
17. Travassos, G.H., Rocha, A.R.C., Um Modelo para Construção e Integração de Ferramentas. In: VIII Simpósio Brasileiro de Engenharia de Software - SBC, 1994, Curitiba - PR.