Gerência de Documentos XML no GOA

Marta Mattoso, Maria Cláudia Cavalcanti, Robson Pinheiro,
Humberto Vieira, Leonardo Guerreiro Azevedo, Carlete Ferreira Marques,
Rodrigo Salvador Monteiro, Fátima Cristina Gonçalves, Cláudia Werner
COPPE/UFRJ - Engenharia de Sistemas e Computação
Caixa Postal 68511 – Rio de Janeiro, 21945-970 - Brasil
goa@dbd.cos.ufrj.br http://www.cos.ufrj.br/~goa

Resumo

O crescente uso de XML para descrever e representar componentes de software motivou o desenvolvimento da gerência de documentos XML no servidor de objetos GOA. Neste artigo, apresentamos a API Cliente GOA XML, desenvolvida para permitir o armazenamento e a manipulação de documentos XML no GOA.

Abstract

The emerging use of XML to describe and represent software components has motivated the development of XML documents management in the GOA object server. In this work, we present the GOA XML Client API, which was developed to store and manipulate XML documents in the GOA server.

1. Introdução

Atualmente, para promover ainda mais a reutilização de componentes gerados por ambientes de Desenvolvimento Baseado em Componentes (DBC), há um forte movimento no sentido de facilitar a interoperabilidade entre estes componentes, tanto no nível operacional, quanto no nível descritivo. No nível operacional, os componentes necessitam de mecanismos padrão de integração para interagir diretamente entre si. Já no nível descritivo, os componentes precisam contar com um formato padrão de descrição de suas características, funcionalidades, etc. No contexto da Web, para evitar o fracasso de soluções proprietárias, a indústria de software se uniu em torno de uma única proposta no sentido de padronizar a interação entre componentes de software conhecidos como Web Services [1], em ambos os níveis citados. Desde 2001, os grupos de trabalho da W3C vêm trabalhando em especificações padrão do protocolo SOAP (*Simple Object Access Protocol)* [2] e da linguagem WSDL (*Web Service Description Language*) [3], ambos expressos na linguagem padrão XML.

Neste contexto, os repositórios de documentos XML, em especial os de componentes de software, tendem a ganhar projeção, com destaque para os Sistemas de Gerência de Banco de Dados (SGBDs). A busca por maior eficiência no armazenamento e recuperação de documentos XML tem ressuscitado antigas discussões com relação aos modelos de representação. Atualmente, os SGBDs Relacionais são os mais populares no mercado. Entretanto, o modelo relacional não se mostra o mais adequado, pois é deficiente quanto ao poder de representação de estruturas em árvore e de mecanismos de herança, por exemplo. Tais restrições não ocorrem em SGBDs com capacidade de representação do modelo de objetos (SGBDOs). SGBDs Relacionais-Objeto e SGBDs Nativos são alternativas para o armazenamento de documentos XML. Entretanto, devido à similaridade entre o modelo de objetos e o modelo XML, acreditamos que os SGBDOs aproximam-se da melhor solução. Nesse sentido, estamos propondo o uso do servidor de gerência de objetos GOA [4] como uma ferramenta para oferecer persistência e manipulação de documentos XML.

O GOA é um protótipo de SGBDO em constante evolução na COPPE-UFRJ. Em 1998 [5] apresentamos suas funcionalidades e ferramentas. Em 2000 [6] exploramos o uso do GOA como componente de persistência do ambiente de reuso do *Odyssey* [7]. Neste trabalho, apresentamos a API

Cliente GOA XML *enabler* (*Goaxe*), que foi desenvolvida para permitir que o GOA fosse capaz de manipular e armazenar documentos XML. A vantagem principal em utilizar o GOA ao invés de um produto de mercado, seja ele orientado a objetos (OO) ou relacional objeto (RO), consiste em sua arquitetura aberta e flexível. Outro ponto importante do GOA é sua aderência a padrões OO, em particular ao ODMG [8].

A seção 2 apresenta a arquitetura GOA e suas extensões, incluindo as funcionalidades para XML. A seção seguinte detalha como é feito o mapeamento de um documento XML para os objetos armazenados no GOA. Por último, a seção 4 conclui o artigo apresentando os projetos onde o GOA tem sido usado como repositório para documentos XML.

2. Arquitetura GOA XML

A arquitetura GOA XML (Figura 1) foi projetada segundo o modelo Cliente/Servidor. O módulo servidor encapsula os serviços de armazenamento e recuperação de objetos em disco, gerência de objetos em memória, gerência de esquemas e processamento de consultas. O módulo cliente, por sua vez, é dividido em duas camadas: as APIs Cliente e *Goaxe*. A API Cliente encapsula as rotinas de chamada aos serviços oferecidos pelo servidor, como por exemplo: abertura e fechamento de bases; inserção, remoção e consulta de objetos; consulta ao esquema da base; etc. Já a *Goaxe* oferece os serviços de mapeamento entre documentos XML e objetos GOA, segundo o padrão DOM [9] da W3C. A unidade de transferência entre o servidor e o cliente é o objeto. A comunicação entre ambos os módulos é feita via *socket*.

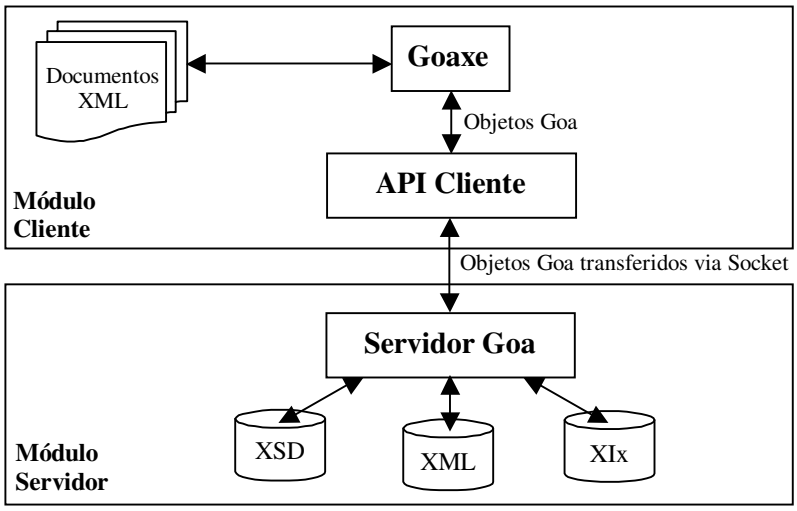

Figura 1 - Arquitetura Goa XML

2.1. O Servidor e a API Cliente GOA

A Figura 2a exibe um resumo do modelo de classes em notação UML do servidor GOA. A classe *GoaSocketCommunication* é responsável por receber via *socket* as requisições dos clientes conectados. A classe *GoaServer* interpreta as requisições solicitadas, repassando-as a classe *GoaDatabase*, que, por sua vez, gerencia o acesso aos dados de uma base em disco através de seus quatro gerentes: (1) o Gerente de Páginas (*GoaPageManager*), responsável pela transferência de páginas do disco para um cache em memória e vice-versa; (2) o Gerente de Objetos (*GoaObjectManager*), responsável pela transferência de objetos das páginas em cache para uma lista de objetos em memória e vice-versa; (3) o Gerente de Esquemas (*GoaSchemaManager*), que gerencia as definições dos elementos do esquema da base, ou seja, as classes (*GoaClassSchemaElement*) e os seus relacionamentos (*GoaRelationshipSchemaElement*); e (4) o Gerente de Consultas (*GoaQueryManager*), que processa consultas no formato OQL (*Object Query Language*) para

recuperação de objetos na base. Todo dado armazenado internamente é representado por um objeto do tipo estruturado (*GoaInstance*), binário (*GoaBlob*) ou coleção (*GoaCollection*). Cada objeto possui uma identidade exclusiva denotada pelo identificador único de objetos ou Oid (*GoaOid*).

Já a Figura 2b exibe o modelo de classes da API Cliente, que é constituída pelas seguintes classes: (1) a classe *GoaDatabase*, que encapsula as chamadas aos serviços oferecidos pelo servidor via socket (*GoaClientSocket*); (2) a classe *GoaDataSet*, responsável pela gerência de coleções de objetos no lado cliente; e (3) a classe *GoaObject*, que armazena as informações de um objeto, incluindo o Oid (*GoaOid*) e seus atributos (*GoaAttribute*) com os respectivos tipos e valores.

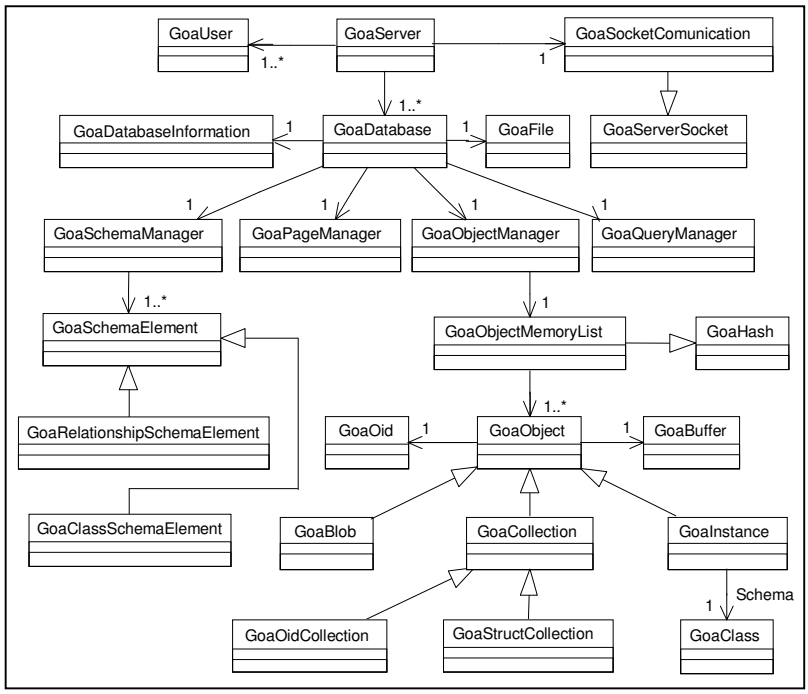

Figura 2a – Modelo de classes do Servidor GOA

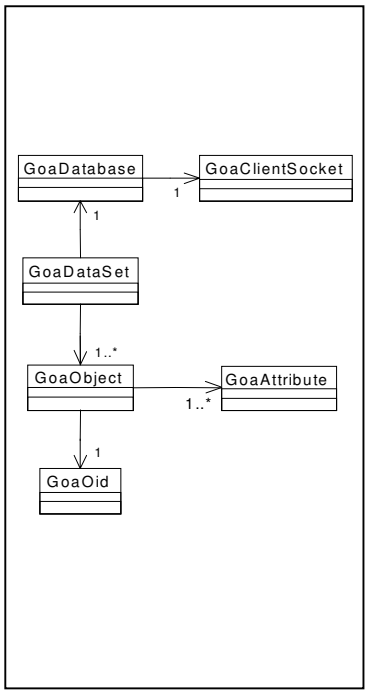

Figura 2b - Modelo de classes da API Cliente GOA

2.2. Extensões GOA para Manipulação de Documentos XML

A extensão GOA para manipulação de documentos XML mantém um único esquema genérico por base, segundo a estrutura DOM, para armazenamento destes documentos, como será detalhado na seção seguinte. Em uma aplicação XML, a manipulação de documentos XML envolve também a validação destes documentos. Para tanto, o repositório de documentos XML precisa armazenar esquemas XML. Estes esquemas são documentos XML e podem, portanto, ser armazenados com base no mesmo esquema de armazenamento usado para o armazenamento de suas instâncias.

Na Figura 1, separamos os documentos XML a armazenar em três bases de dados principais: XML, XSD e XIx. Todas elas são baseadas no mesmo esquema genérico para armazenamento XML. A base de dados XSD armazena esquemas XML usados para validar os documentos XML. A base de dados XML armazena os documentos XML instanciados a partir dos esquemas XML. A base de dados XIx armazena um índice sobre os documentos XML, também em formato XML. Este índice visa agilizar o acesso direto a estes documentos, facilitando buscas por palavras-chave.

Além de prover o mapeamento entre documentos XML e objetos armazenados no GOA, a API *Goaxe* se comunica com a arquitetura *Xverter* [10] responsável por traduzir consultas XQuery para o GOA. Considerando o esquema genérico em que se encontram armazenados os documentos XML, as consultas podem se tornar custosas. Como as classes a percorrer são poucas, o grande volume de objetos concentrados nestas classes pode comprometer o desempenho das consultas. Assim sendo, a

tradução de consultas na *Goaxe* pode se beneficiar do auxílio de um mecanismo de índice que agiliza a consulta aos documentos XML. Este mecanismo encontra-se em desenvolvimento como parte dos resultados de uma dissertação de mestrado. A *Goaxe* também é responsável pela geração e atualização dos índices a serem armazenados pela base de dados XIx.

Uma outra abordagem desenvolvida em paralelo à *Goaxe* pressupõe a existência de um esquema de objetos específico para um dado documento XML. Isto é, ao invés de um esquema genérico, baseado no DOM, um esquema GOA específico é criado de acordo com o conjunto de documentos XML a manipular. Esta abordagem é mais direta que a primeira, na medida em que diminui a distância entre o objeto (documento XML) e sua representação (documento XML armazenado), por outro lado o mapeamento não é automático, dependendo assim do usuário.

3. Mapeamento de um Documento XML para o GOA

No contexto de SGBDs baseados em objetos, acreditamos que o DOM oferece a melhor relação entre a semântica oferecida para representação dos documentos XML e a eficiência de acesso aos dados, visto que utiliza um formato de armazenamento que nos permite acessar um esquema de representação conhecido e reconstruir com facilidade o documento XML. Através da criação e povoamento de classes da árvore DOM no SGBD é possível consultar os dados armazenados. A Figura 3 mostra (1) a representação original do documento Patient.xml e o conteúdo do documento XML armazenado na árvore DOM; (2) as classes DOM; e (3) a representação desse documento como instâncias GOA de classes DOM.

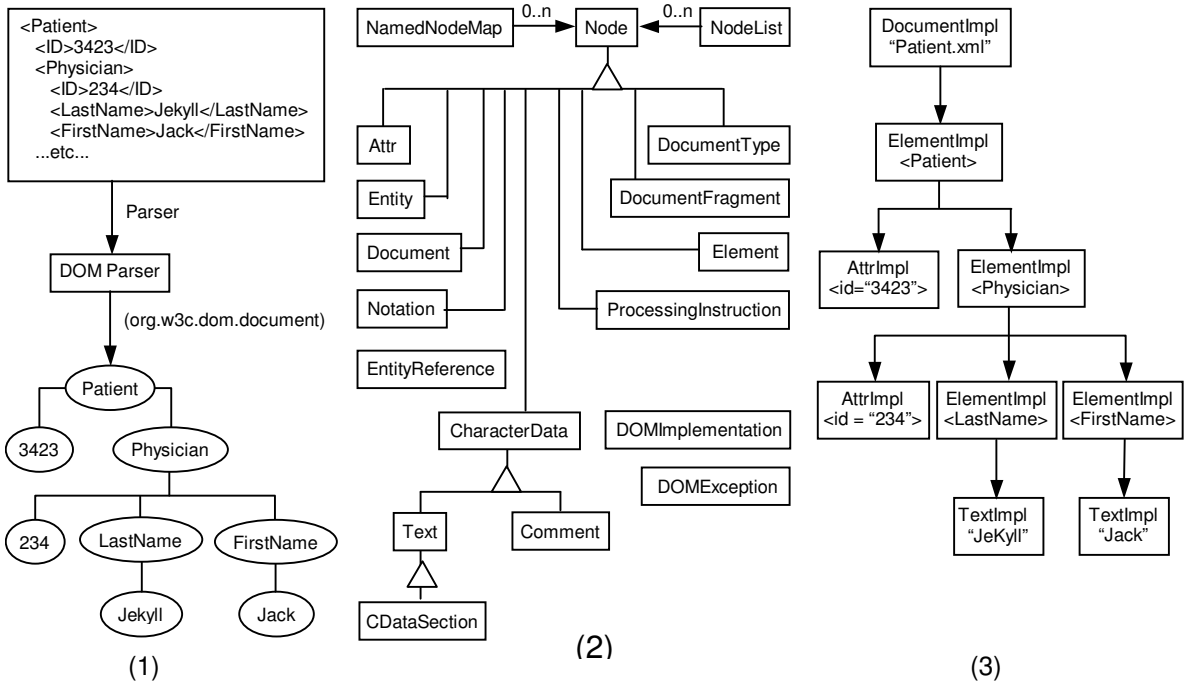

Figura 3 - Mapeamento de um documento XML nas classes da árvore DOM e GOA.

Classes com sufixo *Impl* correspondem à implementação das classes DOM. A classe *DocumentImpl*, como o próprio nome sugere, armazena informações sobre os diferentes documentos XML armazenados na árvore DOM. Já a classe *ElementImpl* é responsável por armazenar os elementos XML, deixando para a classe *TextImpl* guardar o seu conteúdo. A classe *AttrImpl* guarda, além do nome do atributo, o seu valor. A navegação entre as classes é realizada através do atributo *childrenNode*, que pertence à classe *NodeImpl*. Todas as classes citadas anteriormente são herdadas de *NodeImpl*. Os nomes de elementos e atributos XML ficam armazenados no atributo "*name*" e seus

conteúdos no atributo "*value*". As demais classes do W3C DOM se destinam a representar características de documentos XML não presentes no exemplo.

Uma consulta sobre determinado documento, navega sobre classes DOM, começando sobre o *DocumentImpl* e continuando através do atributo *childrenNode*, alcançando desta forma a raiz do documento. Prosseguindo através do *childrenNode*, acessamos os elementos filhos da raiz e assim sucessivamente, até chegarmos aos nós-folhas do documento. É importante ressaltar que os nomes e os valores de atributos XML encontram-se em um mesmo nó da árvore, ao contrário do que ocorre em elementos. Desta forma, para acessar o conteúdo de determinado elemento, precisamos navegar mais uma vez pelo *childrenNode*, para só então recuperar o seu valor.

Para armazenar os objetos da árvore DOM em memória, instanciamos algumas das classes do modelo GOA descrito na Figura 2. Primeiramente, criamos um esquema no GOA conforme mostra o *script* da Figura 4. Este esquema reflete a estrutura completa do W3C DOM contendo todas as suas classes. A classe *GoaSchemaManager*, como o próprio nome sugere, é a responsável por armazenar o esquema de classes no GOA. Uma vez criado o esquema de armazenamento, é necessária a leitura do documento XML para que os objetos correspondentes às classes do DOM sejam criados. O documento é então carregado para a memória para só então começarmos a criar os objetos no GOA. No GOA, a classe *GoaObject* conterá todos os objetos que correspondem aos elementos e dados do documento XML. Portanto, para cada tipo de nó encontrado no documento XML (*ElementNode*, *TextNode*, etc.) é criado um objeto da classe *GoaObject* para armazenar o seu valor. A criação dos objetos termina quando todos os nós do documento tiverem sido percorridos. Na API Goaxe, o documento XML é percorrido em profundidade com o percurso em pré-ordem.

```
CREATE_CLASS Root - Roots;
CREATE_CLASS NamedNodeMapImpl Root NamedNodeMapImpls;
CREATE_CLASS NodeImpl Root NodeImpls name V50 value V50;

CREATE_CLASS NotationImpl NodeImpl NotationImpls;
CREATE_CLASS ProcessingInstructionImpl NodeImpl ProcessingInstructionImpls;
CREATE_CLASS NodeContainerImpl NodeImpl NodeContainerImpls;
CREATE_CLASS CharacterDataImpl NodeImpl CharacterDataImpls;

CREATE_CLASS AttrImpl NodeContainerImpl AttrImpls;
CREATE_CLASS EntityImpl NodeContainerImpl EntityImpls;
CREATE_CLASS DocumentImpl NodeContainerImpl DocumentImpls;
CREATE_CLASS EntityReferenceImpl NodeContainerImpl EntityReferenceImpl;
CREATE_CLASS DocumentTypeImpl NodeContainerImpl DocumentTypeImpls;
CREATE_CLASS DocumentFragmentImpl NodeContainerImpl DocumentFragmentImpls;
CREATE_CLASS ElementImpl NodeContainerImpl ElementImpls;

CREATE_CLASS CommentImpl CharacterDataImpl CommentImpls;
CREATE_CLASS TextImpl CharacterDataImpl TextImpls;

CREATE_CLASS CDataSectionImpl TextImpl CDataSectionImpls;

CREATE_RELATIONSHIP nodes * NamedNodeMapImpl NodeImpl namedNodeMapImpl 1;
CREATE_RELATIONSHIP attrElement 1 NamedNodeMapImpl ElementImpl attributes 1;
CREATE_RELATIONSHIP childrenNode * NodeImpl NodeImpl parentNode 1;
CREATE_RELATIONSHIP docType 1 DocumentImpl DocumentTypeImpl typeDocs 1;
```

Figura 4 - Script de criação do Esquema GOA XML

4. Conclusão

As APIs GOA Client e *Goaxe* foram implementadas em Java. Existe também uma versão da API GOA Client em C++. O Servidor GOA foi implementado em C++ e encontra-se atualmente na versão 3.0. A API *Goaxe* manipula documentos XML através do uso da API DOM Xerces implementada pelo grupo Apache [11], que se baseia no padrão especificado pela W3C (níveis 1 e 2).

As APIs GOA Cliente e *Goaxe* vêm sendo desenvolvidas sob a demanda de alguns projetos e teses. No contexto do projeto *KIWI* [12], propõe-se a utilização do GOA como repositório de

metadados a respeito de recursos científicos publicados na Web, que podem ser dados, programas ou modelos. A idéia é facilitar e monitorar o acesso a estes recursos a partir de consultas a este repositório. Já no contexto do projeto *Odyssey* [13], o GOA é utilizado junto ao componente *ComPublish* [14], na integração de informações em XML de componentes de software publicados em repositórios distribuídos pela Internet.

5. Agradecimentos

Gostaríamos de agradecer ao CNPq e à Faperj pelo apoio dado ao desenvolvimento das extensões do GOA, em especial ao projeto CNPq/CT-Petro número 467.027/00-5. Estendemos o agradecimento a todos os alunos da Linha de Banco de Dados da COPPE que vêm dedicando um grande esforço na consolidação e nas diversas extensões do GOA.

Referências

1. W3C Web Services - http://www.w3.org/2002/ws/.
2. W3C SOAP 1.1 – http://www.w3.org/TR/2000/NOTE-SOAP-20000508.
3. W3C WSDL 1.1 – http://www.w3.org/tr/2001/NOTE-wsdl-20010315.
4. Mauro, R.C.; Mattoso. M.L.Q.; et al.; "GOA++: Tecnologia, implementação e extensões aos serviços de gerência de objetos", In: *XII Simp. Bras. de Banco de Dados*, pp.272-286; Fortaleza, Outubro de 1997.
5. Mauro, R.C.; Mattoso. M.L.Q.; "GOA++ e suas Ferramentas", *Anais da 1a. Mostra Brasileira de Software Acadêmico e Comercial do XIII Simp Bras de Banco de Dados*, SBC, pp.83-88; Maringá, Outubro de 1998.
6. Mattoso, M. et al.; "Persistência de Componentes num Ambiente de Reuso", *XIV Simpósio Brasileiro de Engenharia de Software*, pp.251-254; João Pessoa, Outubro de 2000.
7. Braga, R.M.M.; Werner, C.M.L.; Mattoso, M.L.Q.; "Odyssey: A Reuse Environment based on Domain Models", Anais da IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET99), IEEE CS Press, pp. 50-57; Richardson, Texas, Março de 1999.
8. Cattel, R.G.; Barry, D.K.; "The Object Database Standard: ODMG 2.0 - Morgan Kaufmann Publish"; 1997.
9. W3C DOM – http://www.w3.org/DOM.
10. Vieira, H. Ruberg, G. Mattoso, M. "*Xverter:* Armazenamento e Consulta de Dados XML em SGBDs" , a ser publicado no XVII Simpósio Brasileiro de Banco de Dados, Outubro de 2002.
11. Grupo Apache – http://xml.apache.org.
12. Cavalcanti, M.C.; Mattoso, M.; Campos, M. L.; Simon, E.; Llirbat, F.; "An Architecture for Managing Distributed Scientific Resources", a ser publicado In: 14th International Conference on Scientific and Statistical Database Management, IEEE Computer Society Press; Edimburgo, Julho de 2002.
13. Projeto Odyssey – www.cos.ufrj.br/~odyssey.
14. Souza, R. P.; Costa, M. N.; et. al.; "Software Components Reuse Through Web Search and Retrieval", International Workshop on Information Integration on the Web Technologies and Applications, pp.12-18; Rio de Janeiro, Abril de 2001.