SINS: um Ambiente para Geração de Aplicações baseadas em Serviços

Sérgio Larentis Júnior, Jorge Luis Victória Barbosa, Sérgio Crespo Coelho da Silva Pinto, Andrêsa Vargas Larentis

Programa Interdisciplinar de Pós-Graduação em Computação Aplicada (PIPCA)
Universidade do Vale do Rio dos Sinos (UNISINOS)
Caixa Postal 275 – 93.022-000 – São Leopoldo – RS – Brazil

{sergiolarentis, andresa.vargas}@gmail.com,
{jbarbosa, crespo}@unisinos.br

Abstract. *This article describes a composite application generation environment, called SINS, which combines services available as Web Services to generate composite applications without coding.*

Resumo. *Este artigo descreve um ambiente para geração de composite applications, chamado de SINS, que é capaz de combinar os serviços conhecidos pelo ambiente e que tenham sido desenvolvidos como Web Services em aplicações sem que seja necessária codificação para isso.*

1. Introdução

A era da "industrialização do *software*" está baseada na máxima de que pequenos componentes podem ser feitos de diferentes formas, desde que respeitem os padrões de integração para se obter um resultado final.

Com isso, surge o desafio de normalizar o que poderiam ser esses componentes. Os paradigmas atuais, baseados em objetos, classes ou includes, não conseguem dar a correta idéia de componentização pela falta de interoperabilidade entre plataformas ou até mesmo paradigmas. Fato de fácil comprovação pela atual dificuldade entre integração de componentes em .Net e Java ou de extração de dados de um mainframe para uso em outra aplicação.

Uma forma de tentar resolver esses problemas e possibilitar a analogia com as linhas de produção é o chamado SOA (*Service Oriented Architecture*), que tem como pilares principais a interoperabilidade, a independência de componentes e o reuso. O conceito de desenvolver software pensando nas suas funcionalidades é substituído por pacotes de serviços que possam ser agrupados a fim de permitir o reuso e a escalabilidade das aplicações [TIInside, 2007].

Grandes empresas como Microsoft, Oracle, IBM e outras estão bastante focadas no desenvolvimento de ferramentas que dêem suporte a geração de serviços e gerenciamento, porém, ainda não há um ambiente que organize de forma completa e dinâmica e principalmente, com meios que um usuário possa gerar suas aplicações sem intervenção de um profissional da área de software. Assim, o objetivo do trabalho é desenvolver um ambiente para geração de aplicações baseadas em serviços. Este ambiente é capaz de utilizar serviços existentes em um contexto e disponibilizados em

forma de *Web Service*, catalogá-los para posterior uso e, por fim, prover uma interface ao usuário que lhe permita compor aplicações baseadas nestes serviços.

O artigo está organizado da seguinte forma: A seção 2 apresenta os conceitos de *Web Services*. A seção 3 apresenta o conceito de SOA utilizado à continuação do trabalho, servindo como base para o entendimento da conceituação de serviços. A seção 4 apresenta a idéia do ambiente SINS, modelo, características e limitações. Na seção 5 é apresentada a implementação do ambiente. Na seção 6 são apresentados os principais trabalhos relacionados a este. Por fim, a seção 7 apresenta as conclusões e considerações sobre o trabalho proposto.

2. *Web Services*

Sistemas bem-sucedidos de Tecnologia da Informação (TI) exigem cada vez mais interoperabilidade entre plataformas e serviços flexíveis que possam evoluir facilmente com o tempo. Segundo o *Word Wide Web Consortium* (W3C), a tecnologia de *Web Services* fornece um mecanismo padrão de interoperabilidade entre diferentes aplicações de softwares, executando em uma variedade de plataformas e/ou frameworks [W3C, 2007].

A arquitetura de *Web Services* para disponibilização e acesso aos serviços está baseada nas interações de três papéis: provedor, solicitante e registro de serviço [Kreger, 2001]. A figura 1 apresenta as interações entre esses papéis.

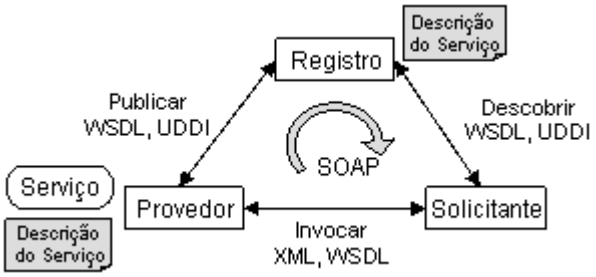

Figura 1. Papéis, operações e artefatos de *Web Services* [Kreger, 2001].

O provedor de serviço é a plataforma acessada na solicitação do serviço. O registro de serviço é o local onde os provedores publicam as descrições dos serviços. O solicitante de serviço é uma aplicação que invoca ou inicia uma interação com um serviço.

As tecnologias padrão utilizadas na construção de *Web Services* são baseadas em XML e permitem invocar um serviço sem a necessidade de conhecer a plataforma ou linguagem de programação usada na sua construção. São elas:

- WSDL (*Web Service Description Language)*: é uma linguagem em formato XML para descrição de interfaces de serviços, de forma que outros programas possam interagir com esses serviços [Hansen, 2003];
- SOAP (*Simple Object Access Protocol)*: permite a comunicação entre diversas aplicações em um ambiente distribuído e descentralizado [Hansen, 2003];

- UDDI (*Universal Description, Discovery and Integration)*: tem como objetivo criar um padrão para a descoberta de serviços e possibilita a localização dos *Web Services* [Newcomer, 2002].

3. *Service Oriented Architecture*

SOA relaciona serviços e seus consumidores, que representam um processo de negócio. Os serviços podem ser acessados pelo nome via uma interface, e os consumidores acessam os serviços disponíveis via interface de serviço, por exemplo, *Web Services* [Natis, 2003].

Com a evolução de XML e *Web Services*, SOA evolui também. Isto é resultado da relação entre numerosas iniciativas dirigidas por uma variedade de organizações de padrões e desenvolvimento de software. Os padrões de *Web Services* continuam a serem adotados em grande número e com isso fornecem suporte ao uso na arquitetura SOA [Rogers and Hendrick, 2005].

Um projeto de *Web Service* para SOA difere de outros *Web Services* criados para uso em outros ambientes, porque segue um conjunto de convenções distintas. Os serviços disponibilizados em uma arquitetura SOA devem ter as seguintes características: reusabilidade, autonomia, *stateless*, passíveis de descoberta e fraco acoplamento. Essas características não são automaticamente fornecidas por *Web Services*, mas sucedem a uma padronização [Erl, 2006].

A figura 2 apresenta um modelo de camadas de uma arquitetura SOA.

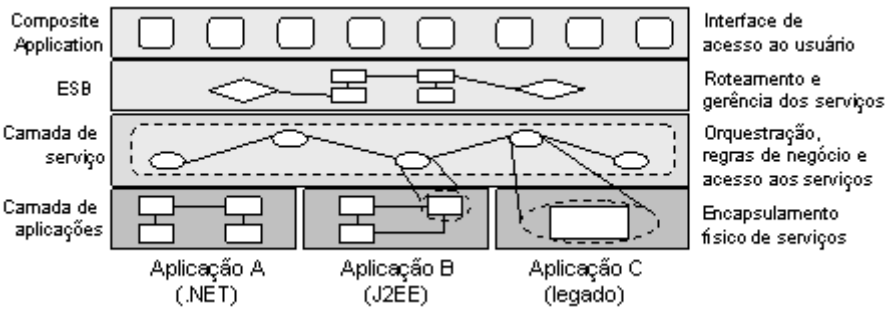

Figura 2. Camadas da arquitetura SOA [Erl, 2006].

Na figura 2, as aplicações A, B e C, pertencentes a uma camada de aplicações, distribuem seus dados como serviço para a camada de serviços, assim, a conexão com essas aplicações deixa de existir, restando apenas os serviços gerados a partir delas. A camada de serviços é responsável por gerenciar os processos de negócios, orquestrar e disponibilizar os serviços para o uso. A camada ESB efetua o roteamento e a gerência dos serviços disponibilizados e a camada de *composite application* fornece uma interface de acesso ao usuário para utilização do serviço requisitado.

3.1 Orquestração de serviços

Orquestração representa o processo pelos quais diferentes serviços são invocados. Os serviços podem ser organizados em diferentes formas ou apenas reagrupados em outros fluxos. A orquestração é composta por um fluxo de etapas, e um coordenador responsável pelo andamento no fluxo. A figura 3 apresenta o processo de orquestração de serviços.

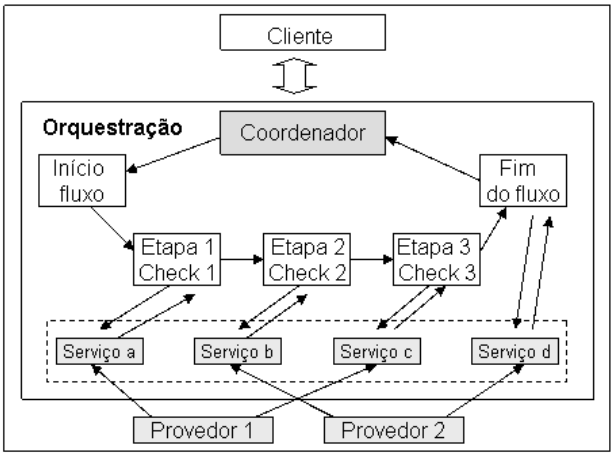

Figura 3. Orquestração de serviços [Sampaio, 2006].

Na figura 3 é apresentado um esquema de orquestração de quatro serviços, fornecidos por dois provedores de serviços diferentes. Neste processo, o cliente se comunica com o coordenador e efetua uma solicitação. O coordenador inicia o fluxo, invocando e verificando todas as etapas necessárias. Cada etapa invoca um serviço, que é fornecido por um provedor de serviço. Desta maneira, podemos mudar a ordem das etapas, acrescentar outras, mudar os critérios de verificação ou criar outros fluxos sem alterar o código dos serviços.

3.2 *Composite Application*

Uma composição refere-se à maneira de disponibilizar soluções na empresa, assemelhando-se a componentes pré-construídos. Isto inclui também habilidades de personalização e de customização, assim as pessoas podem facilmente e rapidamente modificar funcionalidades específicas da solução criada. Os benefícios são substanciais, porque a composição fornece meios para conseguir agilidade, adaptabilidade e alinhamento nos negócios da empresa [Banerjee, 2007].

Uma *composite application* é: "uma coleção de serviços que foram montados para fornecer uma potencialidade ao negócio. Estes serviços são artefatos que podem ser desdobrados independentemente, permitindo a composição e utilização das capacidades de plataformas específicas" [Banerjee, 2007].

Em SOA, *composite application* é o produto final. Estas representam o valor de negócio de uma empresa derivada da sua aplicação de SOA. Independente de a *composite application* ter sido planejada para uso interno ou externo, ela representa como uma empresa pode mapear suas necessidades e processos de negócios para que sejam disponibilizados através dos princípios de SOA

4. SINS

O SINS (acrônimo recursivo de *Sins Is Not SOA*) objetiva criar um ambiente para geração de aplicações baseadas em serviços, buscando gerenciar os serviços de forma a facilitar o uso permitindo que usuários criem aplicativos de forma rápida sem a necessidade de um profissional de software.

O SINS consiste de *Web Services* no papel de serviços, mapeados para atender as necessidades de negócios já implementadas nos sistemas legados (podendo estar em diferentes máquinas, servidores de aplicação e plataforma), ambientados num repositório e comunicando-se via rede.

Os *Web Services* utilizam XML como base de sua comunicação, seguindo o conceito já apresentado por Erl [Erl, 2006] como requisito essencial para comunicação em arquitetura SOA. A necessidade de disponibilizar estes serviços levou à definição do uso de *Web Services*, que seguirão a definição dos princípios da orientação a serviços descritos em Erl [Erl, 2006].

4.1 Arquitetura

O SINS permite uma melhor organização de serviços de forma a facilitar seu uso, e a capacidade de geração de aplicações (*composite applications*) por um usuário, de acordo com suas necessidades, apenas combinando serviços existentes sem a necessidade codificação ou envolvimento de profissionais da área de software;

O ambiente SINS pode ser dividido em duas grandes partes. A primeira delas diz respeito ao ambiente responsável pela orquestração de *Web Services* sob o conceito de serviços de forma a disponibilizá-los na web em forma de uma aplicação (*composite application*).

A outra é uma interface que permite ao usuário consultar um repositório dos serviços conhecidos pelo ambiente e, a partir disso, gerar aplicações a serem disponibilizadas na web sem necessidade de interação de um profissional da área de software. A figura 4 apresenta uma visão da arquitetura do SINS.

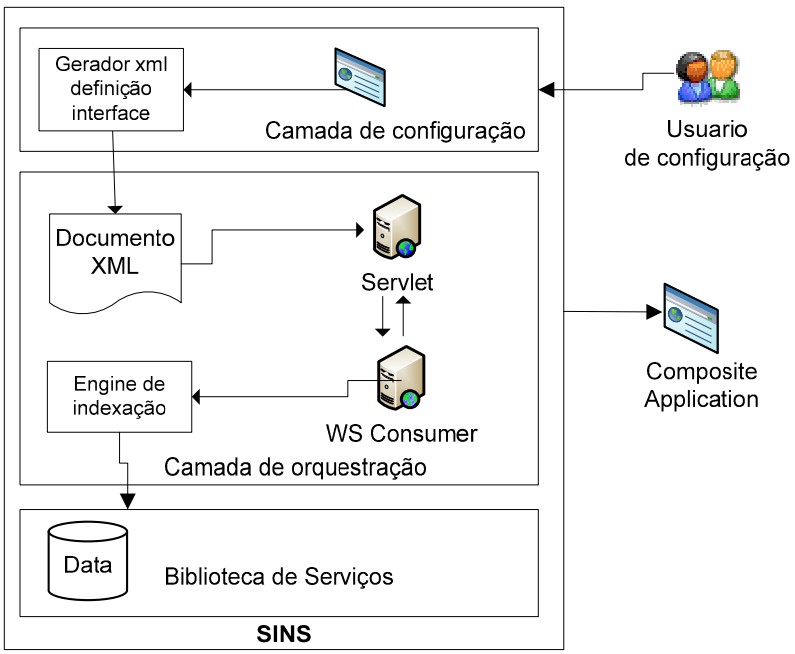

Figura 4. Detalhes das camadas da arquitetura.

É importante destacar os pontos principais da arquitetura do SINS, que são:

- Biblioteca de serviços: são *Web Services* já mapeados e catalogados pelo ambiente que seguem o WSDL padrão deste;

- *Engine* de Indexação: esse motor de indexação percorre o banco de dados que contém as informações sobre os serviços e os relaciona, de forma que possa auxiliar o usuário na posterior confecção de uma aplicação (*composite application*);
- *WS Consumer*: tratam-se apenas de abstrações lógicas para os serviços. É possível que um *WS Consumer* seja um serviço ou que um provider seja n serviços. A idéia funcional dele é redirecionar atividades de um serviço para outro ou combinar vários de forma a obter o mesmo resultado, quando possível, aumentando a disponibilidade da aplicação e auxiliando na tarefa de identificar redundâncias descartáveis nos serviços existentes com intuito de otimização;
- *Servlet*: responsável por processar as requisições e gerar as interfaces das *composite application*;
- Documento XML: documento que armazena os dados que servirão como parâmetros para geração da *composite application*;
- Gerador XML com definição de interface: esse módulo gera um XML a partir das configurações feitas pelo usuário que contém os dados necessários para geração da *composite application*;
- Interface de configuração: fornece as operações para configuração das *composite application*s;
- *Composite application*: é a aplicação em si, que usará os serviços de forma transparente pela camada de orquestração.

4.2 Modelo de Interações

O ambiente SINS permite a configuração para criação de *composite applications* através de uma interface de configuração. O funcionamento do processo de configuração é demonstrado pelo diagrama apresentado na figura 5. A interação ocorre da seguinte forma:

1. O usuário seleciona um ou mais serviços;
2. O usuário realiza as configurações de disposição dos campos ou informações dos parâmetros da aplicação;
3. O usuário configura a URL de acesso a aplicação;
4. O arquivo .xml de configuração é gerado por um processo da arquitetura juntamente com o *Web Service* do serviço configurado, e uma mensagem de confirmação é enviada ao usuário;
5. Os dados são enviados ao repositório de dados;

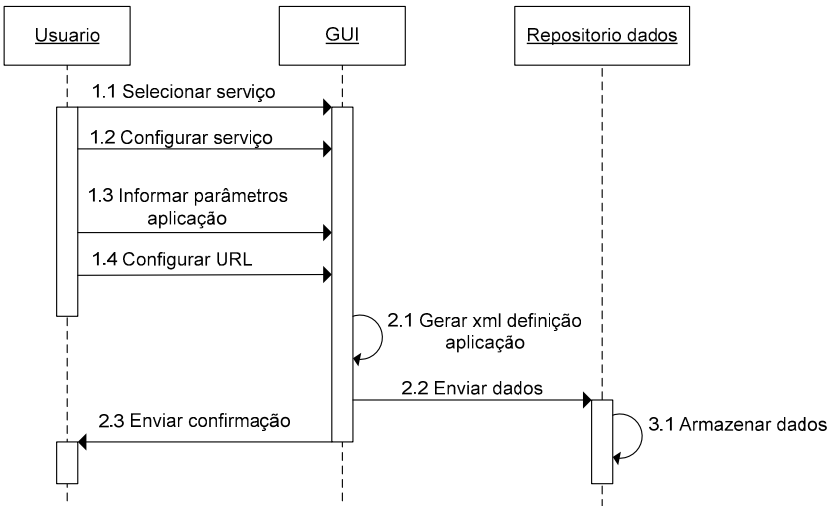

Figura 5. Diagrama de seqüência: camada de configuração.

O principal produto do ambiente SINS é a geração de uma *composite application*. O funcionamento do processo de geração é demonstrado pelo diagrama apresentado na figura 6. A interação ocorre da seguinte forma:

1. O usuário faz a requisição a URL da aplicação;
2. A URL direciona para o *servlet* que criará a *composite application* em tempo real, de acordo com o XML de configuração que é apontado pelo URL da aplicação;
3. Os dados oriundos e destinados ao usuário são trocados diretamente com a aplicação, não sendo mais usada a URL inicial até o fim da comunicação;
4. O *servlet* faz todos os envios e recebimentos de dados com o serviço de modo transparente, sendo a *composite application* a única interface conhecida pelo usuário.

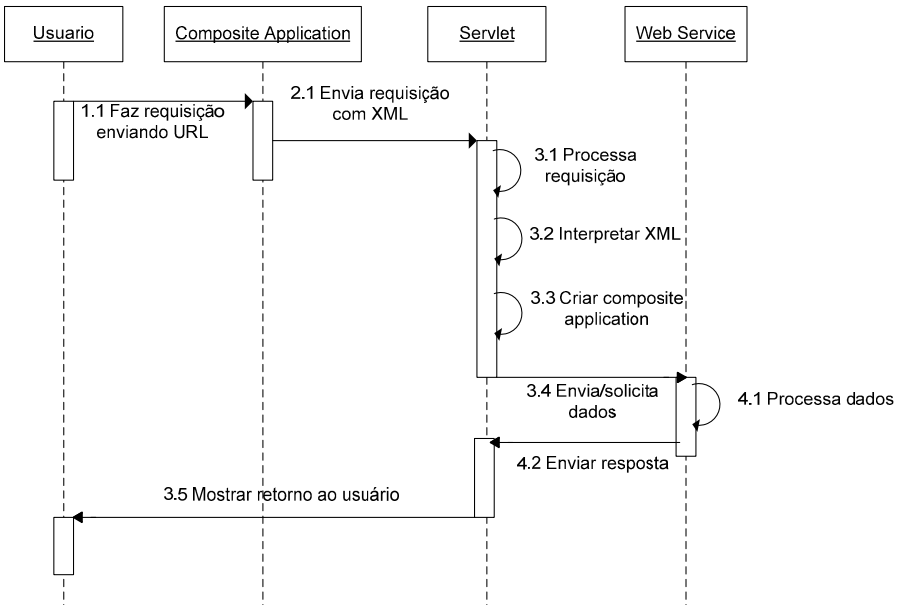

Figura 6. Diagrama de seqüência: camada de orquestração.

O ambiente SINS compõe em sua arquitetura a definição de quatro componentes principais: a *engine* de indexação de serviços e providers, a orquestração, a *composite application* e o repositório de dados, que serão apresentados a seguir:

- *Engine* de indexação de serviços e providers: os serviços são *Web Services* conhecidos pelo ambiente, que estão inseridos no repositório de dados. Os providers tratam-se apenas de um conceito lógico, criados através do resultado do cruzamento de informações de alguns serviços de acordo com sua similaridade, feita pela *engine* de indexação. Isto se fará através dos seguintes critérios: parâmetros de entrada e saída do serviço, comparação sintática da descrição do serviço, comparação sintática do nome do sistema a que o serviço se refere;
- Orquestração: é a parte central do ambiente SINS responsável por efetuar o gerenciamento da biblioteca de serviços, interpretar o arquivo .xml para geração das *composite applications* e ligar uma *composite application* aos seus respectivos serviços. Um *servlet* disponibilizado pelo ambiente é responsável por processar a requisição e gerar a *composite application* para o usuário. A ligação da *composite application* gerada com o respectivo serviço é efetuada através de um *Web Service consumer*. Este trata-se de um middleware que processará as requisições oriundas do arquivo .xml referente a URL enviada;
- *Composite application*: geradas pelo ambiente possuem o mesmo padrão visual, sendo possível apenas pequenas customizações no momento da aplicação através da GUI. Algumas características são destacadas: ela não existe de fato, é criada em tempo real através da leitura do arquivo .xml que contém suas definições; todas as *composite application* possuem o mesmo padrão visual; todas as composite application são geradas pelo mesmo *servlet*, ponto de destaque pelo reuso; regras de negócio, validação e dados são inerentes aos serviços, sendo as *composite application* apenas uma interface para o usuário;
- Repositório de dados: permite armazenar as informações pertinentes a todos os serviços conhecidos pelo ambiente, tais como: sistema a que se refere, descrição do serviço, parâmetros de entrada e saída.

4.3 Características e limitações

O principal produto do SINS é a geração de uma *composite application* e disponibilizá-la para uso.

O SINS é um ambiente bastante focado na geração de *composite applications* sem necessidade de codificação. Devido a arquitetura e objetivo da solução, identificou-se as seguintes limitações:

1. O SINS não possui nenhum mecanismo para otimização de carga ou *pooling*. Tendo em vista que o handler de todas as requisições para geração de *composite applications* é um único *servlet*, pode ser necessário configurar otimizações em nível de servidores ou application server quando as aplicações puderem ser submetidas a um grande número de acessos (*Network load balance* (NLB), *pooling*, *clustering*, etc);

2. O SINS não possui nenhum dispositivo de autenticação nas composite applications geradas, assim como também não é capaz de autenticar-se contra *Web Services* que façam uso de algum dispositivo de segurança ou que utilizem *WS-Security*;
3. Por basear-se no conceito de SOA, o SINS somente é compatível com *Web Services* que tenham sido desenvolvidos sob os conceitos de serviço (principalmente que sejam *stateless* e atômicos), pois ainda que serviços possam ser combinados em uma única interface, eles serão executados de forma individual e sem nenhum mecanismo para controle de sessão.

5. Implementação

O ambiente SINS utiliza tecnologias Java em sua implementação – plataforma de desenvolvimento, linguagem de programação, servidor de aplicação, entre outros. Estas tecnologias oferecem diversos recursos para o trabalho com XML, Interface Web, *Web Services*, banco de dados e independência do sistema operacional utilizado pelo usuário final. No entanto, chama-se a atenção para o fato de que as *composite applications* geradas utilizam um único *servlet* e apresentam o mesmo padrão visual.

O principal objetivo do SINS é permitir a um usuário a criação de *composite applications* através de serviços e, para tal, são disponibilizadas interfaces de mapeamento de serviços e de geração de *composite applications.* A Figura 7 refere-se a tela de criação de *composite applications* que poderão ser invocadas posteriormente para uso.

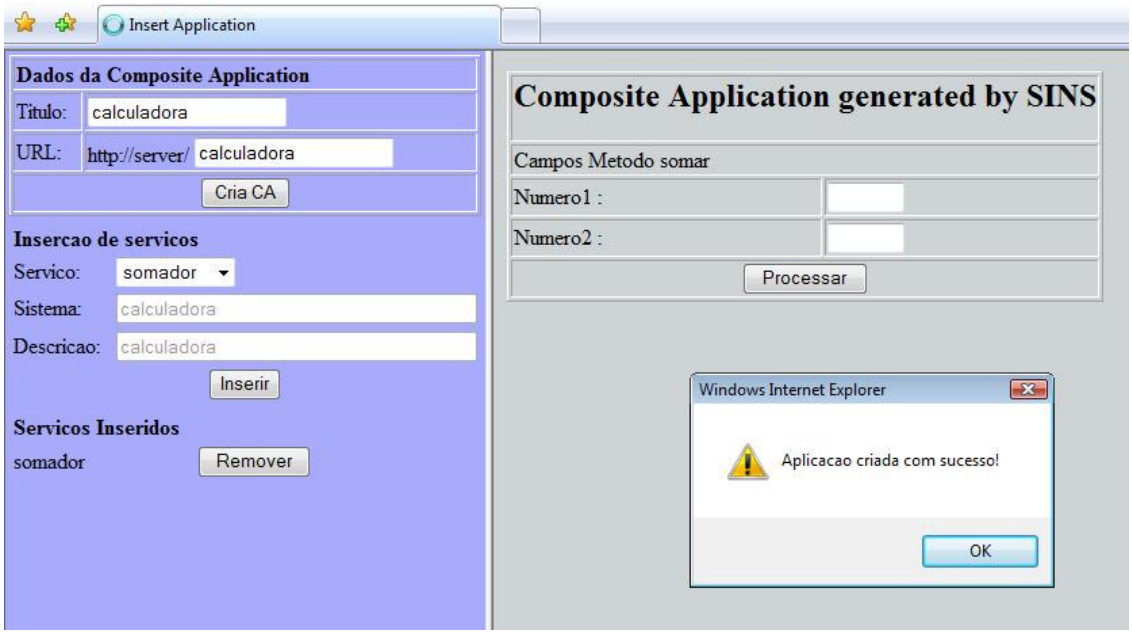

Figura 7. Interface de criação de *composite applications*.

Na parte da esquerda estão os dados de configuração, estando na parte superior os campos de configuração de nome e URL onde a *composite application* deve ser hospedada, enquanto mais abaixo há um combo onde o usuário selecionará todos os serviços que serão inseridos na aplicação. A parte da direita da figura mostra, durante a configuração, a atual diagramação e campos que estão sendo gerados. Uma vez

confirmada a operação, a aplicação estará acessível no endereço digitado no campo URL e terá sua interface e funcionamento tal qual mostrado no *frame* da direita.

As *composite applications* são criadas e armazenadas na estrutura de banco de dados do ambiente para posterior uso, assim como os serviços conhecidos pelo ambiente. O modelo de dados é apresentado na figura 8.

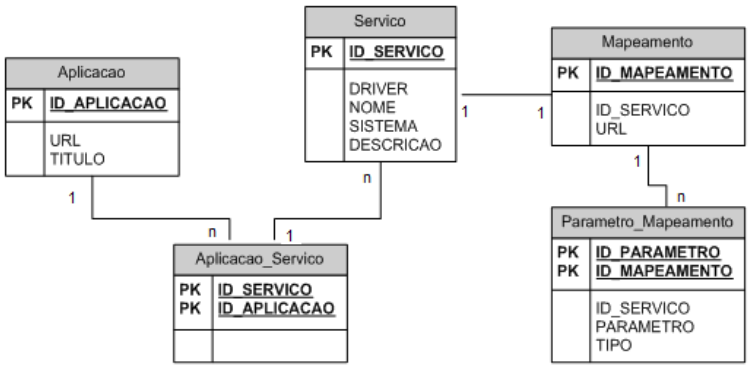

Figura 8. Modelo de dados da biblioteca de serviços.

Após a configuração da *composite application*, seu conteúdo é armazenado no banco de dados e um arquivo xml é gerado contendo sua definição. No momento que o usuário efetua uma requisição ao endereço configurado como referente aquela *composite application*, um *servlet* padrão irá processar a requisição e, através dos parâmetros do arquivo xml irá diagramar a página com os campos e/ou informações enviadas pela *composite application*. O modelo do arquivo xml é apresentado na figura a seguir.

```
<?xml version="1.0">
<dados url="">
<campos>
        <campo></campo>
        <parametro></parametro>
        <tipo></tipo>
        <comprimento></comprimento>
</campos>
<servico>
        <url></url>
        <metodo></metodo>
</servico>
</dados>
```

Figura 9. Modelo do arquivo .xml do interpretador.

Na figura 9 mostra-se um exemplo do XML que o interpretador lerá no momento de gerar uma *composite application*. Nele consta as informações de URL de acesso, os campos que a interface conterá (o conteúdo entre as tags <campos> serão repetidas tantas vezes quantos campos existirem na interface) e, por fim, os campos URL e método, que dizem respeito ao endereço e método do *Web Service* que será usado pela *composite application*.

6. Trabalhos relacionados

Em tempo de pesquisa para o trabalho, nenhuma trabalho ou ferramenta focada na geração *composite applications* sem a necessidade de codificação, principal objetivo e motivação do SINS, foi encontrada. No entanto algumas ferramentas e publicações

relacionadas ao aspecto de desenvolvimento e migração de legados para *composite applications* foram encontradas e uma breve descrição destes trabalhos é apresentada a seguir:

- Link et al. [Link, 2006] desenvolveram um trabalho que tem como foco principal a camada de apresentação de SOA e a interface para o usuário final, não estando no escopo do trabalho a conexão com a camada de processos. A idéia central está em fornecer componentes de apresentação aplicáveis em ambos nova e antiga arquitetura de software. Além disso, utilizam WSPR (*Web Service Remote Portlets*) na camada de apresentação de SOA, que torna possível a integração com portlets de diferentes provedores de serviços sem interesse na sua implementação. Um processo é fornecido para estender a aplicação legada e assim permitir a integração e invocação de portlets, com isso, os componentes de apresentação existentes são fornecidos como portlets: identificar os componentes de apresentação reusáveis; duplicar o componente usando tecnologia de portlet; integrar este portlet novamente a aplicação legada.
- Algumas soluções comerciais que fazem uso de codificação para geração de *composite application*, tais como: IBM SOA Foundation[1], Microsoft Biztalk[2], Oracle SOA Suite[3] e SAP NetWeaver[4], oferecem diferentes mecanismos para criação de *composite application*, porém, necessitam de codificação para criação da interface, independente da linguagem e fazendo uso de *Web Services*. O SINS faz isso de forma dinâmica e através de uma interface que permite a execução desta tarefa sem a necessidade de escrita de nenhuma linha de código e de forma acessível a uma pessoa que não seja da área de software.

7. Conclusão

Algumas soluções foram encontradas no decorrer deste trabalho e relatadas na seção de trabalhos relacionados. Muitas destas soluções trabalham com a idéia codificação para criação da interface, independente da linguagem, fazendo uso de *Web Services* (com uma pequena melhora para a ferramenta da Oracle, que possui um Wizard para essa tarefa, porém ainda exigindo codificação).

O ambiente SINS gera *composite applications* de forma dinâmica e através de uma interface que permite a execução desta tarefa sem a necessidade de escrita de código, o que difere de outras ferramentas existentes como comentado no parágrafo anterior.

Entretanto, o ambiente SINS não permite a geração de interfaces baseadas em um modelo gráfico que organize melhor as informações na tela e customizadas por usuário, e também não possui capacidade para exportação de *composite applications* seguindo os padrões de portlets. Estas melhorias fazem parte de alguns trabalhos futuros sugeridos para o ambiente SINS.

---

[1] IBM SOA Foundation – http://www.ibm.com

[2] Microsoft BizTalk – http://www.microsoft.com

[3] Oracle SOA Suite – http://www.oracle.com

[4] SAP NetWeaver – http://www.sap.com

Referências

Banerjee, A. (2007) “Building Office Business Applications”. The architecture journal. Input for Better Outcomes. Journal 10. Microsoft. ARC 098-107185.

Erl, T. (2006) Service Oriented Architecture: Concepts, Technology, and Design, Prentice Hall.

Natis, Y. V. (2003) Gartner Research: Service-Oriented Architecture Scenario, ID Number: AV-19-6751, April.

Hansen, R. P. (2003) “GlueScript: uma linguagem específica de domínio para composição de Web Services”. Dissertação de Mestrado – Universidade do Vale do Rio dos Sinos, Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, São Leopoldo.

Kreger, H. (2001) “Web Services Conceptual Architecture”, IBM Software Group, May.

Link, S., Jakobs, F., Neer, L., Abeck, S. (2006) “Architecture of and Migration to SOA´s Presentation Layer”, C&M Research Report, Universität Karlsruhe.

Newcomer, E. (2002) “Understanding Web Services”, Independent Technology Guides, David Chappell, Series Editor.

Revista TIInside Ano 3, Número 22, Março de 2007. Página 18 a 22, http://www.tiinside.com.br/.

Rogers, S. and Hendrick, S. D. (2005) “Oracle Builds Comprehensive SOA Platform”, IDC White Paper, January.

Sampaio, C. (2006) “SOA e Web Services em Java”. Rio de Janeiro: Editora Brasport.

Web Services architecture (2007): “W3C working group note”, http://www.w3.org/TR/ws-arch, February.