UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

Relatórios Técnicos
do Departamento de Informática Aplicada
da UNIRIO
nº 0010/2010

UM ESTUDO SOBRE MASHUP PARA O DESENVOLVIMENTO DE APLICAÇÕES EM UMA ABORDAGEM SOA

Flávio de Paula Faria
Leonardo Guerreiro Azevedo

Departamento de Informática Aplicada

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
Av. Pasteur, 458, Urca - CEP 22290-240
RIO DE JANEIRO – BRASIL

Relatórios Técnicos do DIA/UNIRIO, No. 0010/2010
Editor: Prof. Sean W. M. Siqueira Setembro, 2010

Um Estudo sobre Mashup para o Desenvolvimento de Aplicações em uma Abordagem SOA

Flávio de Paula Faria, Leonardo Guerreiro Azevedo
Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)
Departamento de Informática Aplicada (DIA) – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

{flavio.faria, azevedo}@uniriotec.br

Abstract. The Web 2.0 evolution brought some challenges for the development of *ad hoc* applications (also called situational applications) in order to achieve specific business goals. These applications are typically built with information from heterogeneous and distributed data sources. In this scenario, the SOA approach introduces the service composition concept. Composition aims at reusing existing services for the development of new applications. Mashup extend the composition characteristics to simplify the development of *ad hoc* applications. This report presents the key concepts for the development of mashup applications.

Keywords: SOA, Mashup, Service composition, Data integration.

Resumo. A evolução da Web 2.0 trouxe alguns desafios para o desenvolvimento de aplicações *ad hoc* (conhecidas também como aplicações situacionais) com objetivo de atingir uma meta específica de negócio. Esses aplicativos são normalmente construídos com a integração de informações de fontes de dados heterogêneas e distribuídas. Neste cenário a abordagem SOA apresenta o conceito de composição de serviços, com a finalidade de reutilizar serviços existentes para o desenvolvimento de novos aplicativos. A proposta da composição de serviços *mashup* é estender as características de composição apresentadas em SOA com o objetivo de simplificar o desenvolvimento de aplicações *ad hoc*. Este relatório apresenta os principais conceitos para a desenvolvimento de aplicações *mashup*.

Palavras-chave: SOA, Mashup, Composição de serviços, Integração de dados.

Sumário

1. Introdução 5
2. Principais Conceitos 6
   2.1 Composição de Serviços e m*ashups* 6
   2.2 *Mashup* e a integração de dados 8
   2.3 Arquitetura *mashup* 9
3. Desenvolvimento *mashup* 11
4. Exemplos de Uso 15
5. Conclusões 17
6. Referências Bibliográficas 17

Figuras

Figura 1 - Arquitetura de Padrões de Integração *mashup* segundo Liu *et al.* [2009]. 10
Figura 2 - Arquitetura *mashup* 10
Figura 3 - Consulta de imóveis em Miami utilizando o *HousingMaps*.....16

1. Introdução

O desenvolvimento de sistemas distribuídos que combinam informações a partir de fontes de dados heterogêneas, com o objetivo de compor uma única aplicação, tem cada vez mais sido adotado pelas organizações. Busca-se flexibilidade e reuso para atender às necessidades do negócio. Segundo Josuttis [2007] os processos e sistemas corporativos estão se tornando cada vez mais complexos. Neste cenário, sistemas que antes eram locais começam a se tornar distribuídos. A manutebilidade é um dos principais desafios. Ele ressalta que a forma tradicional de lidar com problemas de escalabilidade e distribuição não é adequada, por isso uma nova abordagem que permita que os sistemas sejam flexíveis e escaláveis é necessária, uma dessas abordagens é SOA.

A Arquitetura Orientada a Serviços (SOA) é um paradigma para a realização e manutenção de processos de negócio em um grande ambiente de sistemas distribuídos que são controlados por diferentes proprietários [Josuttis, 2007].

A complexidade na implantação da arquitetura SOA é um consenso, pois abrange atividades diversas como a análise de softwares de apoio (barramentos de mensagens ou *Enterprise Service Bus* (ESB) [Hewitt, 2007], servidores etc.), integração de aplicações com processos de negócio, definição de processos para modelagem, composição, projeto, monitoramento e gestão dos serviços [Papazoglou *et al*., 2007].

Uma das idéias centrais de SOA é permitir que aplicações sejam projetadas e disponibilizadas como serviços. SOA é um estilo de arquitetura onde funcionalidades de software são disponibilizadas como serviços, facilitando assim a comunicação e interoperabilidade entre aplicações, além de criar uma infra-estrutura de TI mais flexível e alinhada com o negócio [Erl, 2005].

Em uma Arquitetura Orientada a Serviços, Papazoglou *et al.* [2007] define que serviços são elementos computacionais auto-contidos, independentes de plataforma que suportam composição fácil, rápida e de baixo custo para aplicações distribuídas, mantendo baixo acoplamento. Segundo Erl [2005], a principal tecnologia para implementação desses serviços é *web services*.

Zou *et al.* [2007] afirma que os serviços *mashup* são a grande promessa para uma solução SOA flexível no cenário corporativo.

De acordo com Bozzon *et. al* [2009] *mashup* é a composição de funcionalidades existentes, dados e interfaces permitindo alcançar um objetivo mais complexo. Os dados são integrados de fontes de dados heterogêneas, normalmente como *web services,* e o resultado da integração de todas as invocações aos serviços são apresentadas em um *web site* com bons recursos de usabilidade.

Mashup apresenta-se como uma alternativa para integração de informações, a partir do desenvolvimento de aplicações orientadas a serviço, para o atendimento de necessidades *ad-hoc*, que alguns autores classificam como aplicações situacionais, ou seja, aplicações que são desenvolvidas para situações ocasionais de negócio [Jhingran, 2006; Maximilien *et al.,* 2007]. Para Watts [2007] *mashup* é um tipo de aplicação situacional, composta por dois ou mais componentes distintos, interligados para criar uma nova experiência integrada.

A construção de aplicações *mashup* é realizada através da composição de serviços. Composição de serviço envolve papéis (*roles*) e funcionalidades para agregar múltiplos serviços em um único serviço composto [Papazoglou *et al.*, 2007].

O objetivo desse trabalho é apresentar as diferenças entre *mashup* e as formas tradicionais de composição, quais as vantagens e desvantagens dessa abordagem. Serão apresentadas propostas de arquitetura, abordagens de desenvolvimento e um exemplo de uso de uma aplicação *mashup*, bem como os detalhes técnicos para a construção do aplicativo em três diferentes ferramentas.

Este relatório esta organizado nas seguintes seções. A seção 1 é a presente introdução. A seção 2 apresenta os principais conceitos relacionados a este trabalho. Na seção 3 apresentaremos questões referentes ao desenvolvimento de *mashups e* algumas ferramentas. Nas seções 4 e 5 são apresentadas, respectivamente, exemplos de uso e as conclusões obtidas no decorrer do estudo realizado durante o trabalho, e por fim, no capítulo 6, as referências bibliográficas.

2. Principais Conceitos

Esta seção apresenta os principais conceitos do trabalho.

2.1 Composição de Serviços e m*ashups*

Papazoglou *et al*. [2007] apresenta os desafios da composição de serviço e define composição de serviços como a criação de um novo serviço a partir de serviços existentes. As técnicas mais utilizadas para composição de serviços são orquestração e coreografia. Orquestração é uma abordagem baseada em modelo de *workflow*, contendo nós (*web services*) e o fluxo de dados entre eles. Isto significa que a *engine* interpreta o modelo de orquestração e invoca os *web services* com os parâmetros corretos no momento certo. Os pontos positivos dessa abordagem são a disponibilidade e a conformidade com padrões de implementação, como, por exemplo, WS-BPEL [BPEL, 2008]. Outra vantagem é que uma aplicação é definida em um único local, no modelo *workflow*, abstraindo detalhes técnicos de implementação. O fato do modelo ser implementado por uma única máquina facilita a implementação, administração e monitoramento da orquestração, no entanto essa abordagem possui limitações referentes à escalabilidade. Na coreografia a colaboração entre *web services* é descentralizada e um conjunto de mensagens é trocado entre os serviços. Nessa abordagem, um *web service* se comunica diretamente com outro *web service*, ao invés de se comunicar com uma máquina central, resultando em aumento de escalabilidade. Por outro lado, modelar uma aplicação utilizando essa abordagem é difícil e os padrões utilizados são complexos de entender e usar. Do ponto de vista de implementação a coreografia é problemática. Cada máquina é responsável por rotear mensagens e implica em publicar código em todas as máquinas envolvidas, o que nem sempre é possível. Problemas decorrentes da execução são difíceis de gerenciar e a localização dinâmica de um serviço não é fácil de realizar [Pedraza e Estublier, 2009].

Para alguns autores, a composição de serviço pode ser simplificada com o uso de *mashup*. De acordo com Liu *et al.* [2007], *mashup* introduz uma abordagem mais simples e econômica reduzindo a complexidade da composição de serviço em SOA. Existe um formalismo que aplicações compostas são normalmente baseadas em abordagens basadas em *web services* SOAP, utilizando BPEL [BPEL, 2008] e são desenvolvidas por programadores profissionais. Essas abordagens de composição são centradas no nível de *interface* e não no nível de aplicação, o que significa que o usuário deve ter conhecimento das *interfaces* dos serviços para utilizá-los. Por outro lado, Liu *et al.* [2007] a-

ponta que *mashup* utiliza técnicas simples para a composição de serviço e possui as seguintes características:

- Maior reuso – comparando com as tecnologias de composição de serviços SOA, como BPEL e WSCI [WSCI, 2002], *mashup* possui maior granularidade, pois está centrado no nível de aplicação. Cada "bloco" *mashup* possui o seu próprio contexto e lógica de negócio, sendo uma combinação de dados, processo e interface do usuário, logo os serviços *mashup* possuem maior grau de reuso.

- Baseado na web – *mashup* é baseado em tecnologias *web* como *Javascript*, *web services* e *feeds,* utilizadas sobre protocolo HTTP, e JSON (*Javascript Object Notation*) para manipulação de dados. Iniciativas como *OpenAJAX* podem nos levar a convenções de interoperabilidade (padronização) entre os componentes desenvolvidos e os navegadores (*browser*).

- Leve (*lightweight*) – *mashup* utiliza serviços de dados de *sites* públicos e aplicações. *Mashup* é simples de implementar e pode ser construído com uma quantidade pequena de código. O principal benefício está em satisfazer as necessidades dos usuários reduzindo o custo de desenvolvimento.

- Centrado no consumidor final – *mashup* deve suportar um modelo de programação voltado para o usuário final (não para desenvolvedores) em ambientes simples. Todo usuário deve ser capaz de criar sua própria composição de serviços arrastando e soltando artefatos em um navegador *web.*

Essas características se comparada às formas tradicionais de composição, voltadas para desenvolvedores, como BPEL e WSCI, pode tornar a criação de novos processos e o desenvolvimento de novas soluções mais ágil e mais barato. Inclusive essas tecnologias, apesar de endereçarem os problemas de SOA nas organizações, possuem uma curva de aprendizado alta, não permitem composição *on the fly* e não oferecem suporte à composição de aplicações legadas que não possuem integração com *web services.* Segundo Liu *et al.* [2007], essas questões podem comprometer a adoção de SOA, como promessa do paradigma Web 2.0 (simples, rápido, sem barreiras e centrado no usuário). Com isso surge a necessidade de novas tecnologias de composição de serviço, incluindo o modelo de programação e ferramentas, que possam endereçar as necessidades de usuários com pouco conhecimento de programação, e que permita composição de forma flexível e customizável de serviços, dados e aplicações nas organizações ou na *web*. O desafio é que essas tecnologias utilizem a *web* como ferramenta para desenvolvimento e execução (*runtime*) da composição de serviço. Além disso, devem permitir customização e implantação (*deployment) on the fly* da composição para se adaptar mais facilmente às mudanças de negócio. Na opinião de Liu *et al.* [2007], *mashup* é extremamente centrada no consumidor final e é uma tecnologia mais leve para composição de serviços (*lightweight*), aplicável a todos os usuários na internet em um paradigma *Web* 2.0. *Mashup* reduz drasticamente o custo de desenvolvimento de aplicações porque o próprio usuário final pode criar, testar e compartilhar um *mashup*.

Para Gartner [2006] *mashup* é uma estratégia leve (*lightweight*) para integração a partir de dados e serviços de *sites* e aplicações públicas. A implementação é realizada com uma quantidade menor de código. O principal benefício é a redução do tempo e do custo de desenvolvimento de soluções e conseqüentemente a satisfação do usuário. Por outro lado, por combinar dados e lógica de diferentes fontes, *mashup* fica vulnerável a possíveis falhas que possam ocorrer em qualquer uma dessas fontes.

Yu *et al*. [2008] afirmam que *mashup* facilita a composição de serviços para o usuário final e que em geral são simples e fáceis de usar e acessar. Com o uso de ferramentas e alguns componentes disponíveis, usuários estão aptos a compor seus próprios *mashups*.

Bozzon *et. al* [2009] define *mashup* como sendo a composição de funcionalidades existentes, dados e interfaces permitindo alcançar um objetivo mais complexo. Os autores propuseram um método para especificar a composição dos serviços *mashup*. A técnica utiliza BPMN[1] para descrever a visão de alto nível de um *mashup*. Através de transformações de modelos, são gerados os modelos WebML[2] (*Web Modeling Language*). A principal contribuição da proposta consiste em definir um conjunto de linguagens de especificação padrão e transformações de modelos que enderecem todos os aspectos de projeto para o desenvolvimento *mashup*, juntamente com a especificação de um processo de desenvolvimento. A utilização de MDD possibilita a transformação de modelos BPMN em modelos WebML mais detalhados.

2.2 *Mashup* e a integração de dados

Aplicações *mashup* são caracterizadas por integrar dados de fontes de dados distintas para compor uma nova aplicação. O paradigma *mashup* leva em consideração os objetos envolvidos na integração e como esses objetos são compostos, ou seja, a lógica de composição [Yu e Casati, 2008].

Por exemplo, para a construção de uma aplicação *mashup* que liste todos os filmes que tiveram a participação dos vencedores do Oscar no último ano, será necessário realizar uma operação de junção (*join*) entre os nomes dos vencedores e um banco de dados de filmes para retornar a informação desejada [Tuchinda *et al.* 2008]. Liu *et al.* [2007] afirma que um dos maiores desafios no uso de *mashups*, e talvez um dos maiores problemas que os desenvolvedores dessas soluções terão que enfrentar, está relacionado à integração de dados.

Por se tratar de um conceito relativamente novo, pode ser confuso distinguir *mashup* da composição de serviços tradicionais [Liu *et al.* 2007]. Essencialmente *mashup* utiliza a abordagem de composição de serviço utilizada em SOA. No entanto, a composição de serviços *mashup* é uma tecnologia *ad hoc* que permite aos usuários, de diferentes níveis, utilizarem informações de fontes de dados externas para a criação de novos serviços que oferecem novas funções para o usuário. Segundo Hoyer *et al.* [2008] *mashup* é um estilo de composição leve que prega o reuso em diferentes contextos.

Mashup é essencialmente uma técnica que integra dados e conteúdo de diferentes fontes na internet oferecendo uma nova função para o usuário. São utilizadas API's, que podem ser reutilizadas através de *interfaces* públicas para criar novas aplicações. Sobre uma perspectiva SOA, é uma nova maneira de compor serviço, que estende os conceitos de composição apresentados por SOA propondo uma forma mais flexível de composição.

[1] http://www.bpmn.org/

[2] http://www.webml.org/webml/page1.do

2.3 Arquitetura *mashup*

Para integrar informações de diversas fontes de dados e criar um único serviço ou aplicação, a arquitetura de integração deve estar em conformidade com as questões referentes à integração, gerenciamento, transformação e organização dos dados.

Segundo Liu *et al.* [2009], as arquiteturas corporativas possuem outros desafios, como, por exemplo, a integração com aplicações legadas. Além disso, elas devem ser baseadas em uma arquitetura de componentes reutilizáveis, facilitando a adaptação e desenvolvimento das aplicações *mashup*. A proposta apresentada pelos autores é o desenvolvimento de uma arquitetura baseada em padrões de integração, contendo os componentes chave para o desenvolvimento de aplicações *mashup*.

A arquitetura proposta por Liu *et al.* [2009] possui três aspectos centrais: fluxo de dados, integração de dados e *interfaces*. Ela é baseada em padrões que se referem às boas práticas para o desenvolvimento de sistemas corporativos. Contudo, vale ressaltar que os padrões sofreram adaptações para atender às necessidades das arquiteturas *mashup*, que possui necessidades básicas na integração entre dados e funções. Uma vez que os padrões são independentes de plataforma, eles podem ser mapeados para os componentes *mashup*. Entretanto, quando múltilpos padrões são combinados, a estrutura dos componentes *mashup* deve ser cuidadosamente projetada nos pontos de interseção entre os padrões. A seguir é apresentado um resumo dos padrões utilizados pela arquitetura apresentado por Liu *et al.* [2009].

- *Piper* e *Filter*: Padrão responsável pela lógica de integração dos dados. Consiste em uma série de componentes interligados, cada um com uma função específica, como por exemplo, filtrar dados, juntar registros (*join*) ou dividir um fluxo de informação (*fork*). A vantagem em utilizar este padrão é a simplicidade em modelar o fluxo de dados e a facilidade de manutenção, uma vez que os *mashups* são executados da mesma forma com que são projetados em tempo de execução.

- *Data Federation:* Padrão responsável pela integração dos dados estruturados e não estruturados de diversos *datasources*. Originalmente, o padrão não permite redundância de dados, mas no contexto dos *mashups* algum nível de redundância é necessário para simplificar o projeto dos *mashups*. Os dados são gerados pelo *Piper* e *Filter* em diversos formatos para serem processados pelo componente *mashup* e podem ser armazenados em um banco de dados e reutilizados a cada vez que são requisitados.

- MVC: No padrão MVC o modelo acessa a camada de dados da aplicação. O controlador, por sua vez, baseado em uma requisição do usuário se comunica com o modelo e localiza a visão responsável pela apresentação dos dados (por exemplo, HTML, XML, *JavaScript*) para o usuário (geralmente em um *browser*). Na solução apresentada, a camada de modelo implementa o padrão *Data Federation*. O modelo é conectado aos componentes *mashup,* que são os controladores, e implementam o padrão *Pipes and Filter*. O controlador inclui o servidor *web* responsável pela apresentação do conteúdo HTML ou XML para o navegador. Segundo os autores existem dois caminhos para utilizar MVC em aplicações *mashup*, chamados *Server Side Mashup e Client Side Mashup*.

No modelo *Client Side*, a lógica de composição da camada de apresentação é realizada no cliente através de *JavaScript*. Nesta abordagem, podem ocorrer limitações para lógicas complexas. Na abordagem *Server Side*, a integração dos dados que com-

põem a visão ocorre no servidor *web*. Segundo os autores a abordagem Server Side oferece maior escalabilidade se combinada aos padrões *Data Federation* e *Pipes and Filter*.

A Figura 1 apresenta o nível conceitual da arquitetura *mashup* e a integração entre os padrões *Piper* e *Filter, Data Federation e* MVC.

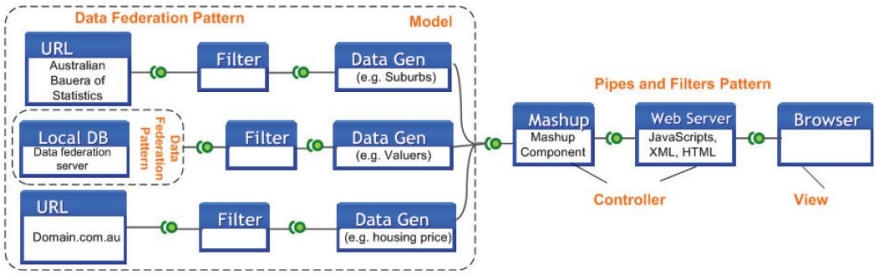

Figura 1 - Arquitetura de Padrões de Integração *mashup* segundo Liu *et al.* [2009]

De acordo com Merrill [2009] a arquitetura de uma aplicação *mashup* (Figura 2) é muito similar à popular arquitetura de três camadas. É composta de três diferentes participantes que estão lógica e fisicamente separados (pelas fronteiras das organizações). Os três participantes são: API/provedores de conteúdo, site *mashup*, *web browser* cliente.

- API/provedores de conteúdo - são os provedores que serão utilizados pelo *mashup*. Para facilitar a consulta aos dados, os provedores expõem seus conteúdos através de protocolos como REST, *Web Services* e RSS/*Atom*.

- Site *mashup* – local onde o *mashup* é hospedado. É onde a lógica do *mashup* está implementada, mas não necessariamente onde o *mashup* é executado. *Mashups* podem ser implementados como aplicações web tradicionais, utilizando um servidor para a geração dinâmica de conteúdo (com tecnologias como Java, CGI, PHP ou ASP), ou diretamente no *browser* do cliente (*Javascript* ou *Applets*). Os benefícios da implementação da lógica *mashup* no cliente é a diminuição da carga do servidor, pois os dados podem ser recuperados direto do provedor do conteúdo, e a atualização das páginas ocorre em partes, de acordo com o seu conteúdo, sem que haja necessidade de atualizar a página inteira. O *Google Maps* API é um exemplo dessa abordagem. Outra abordagem utilizada é a combinação da implementação no servidor e cliente.

- *Web browser* cliente – meio pelo qual o usuário interage com o *mashup*.

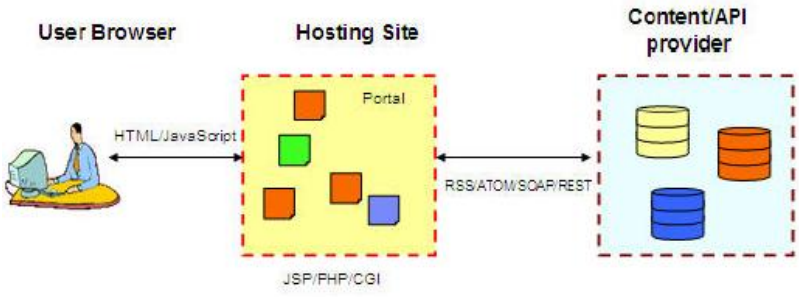

Figura 2 - Arquitetura *mashup*

Apesar da arquitetura *mashup* possuir outros desafios, conforme apresentado anteriormente, ela é baseada em padrões e arquiteturas amplamente utilizadas para o desenvolvimento de software tradicional, como por exemplo o padrão MVC, que se tornou popular com o desenvolvimento de aplicações *web,* e na arquitetura de três camadas com algumas adaptações. Ambas as propostas apresentadas por Liu *et al.* [2009] e Merrill [2009] seguem essa abordagem. No entanto, a proposta apresentada por Liu *et al.* [2009] é baseada em padrões de projetos e apresenta um nível de detalhamento maior do que a apresentada por Merrill [2009]. Liu *et al.* [2009] endereçam ainda a questão da integração de dados, fundamental para o desenvolvimento de *mashups,* o que não é tratado por Merrill [2009].

3. Desenvolvimento *mashup*

Para Yu *et al.* [2008] o desenvolvimento *mashup* difere do desenvolvimento tradicional de aplicações baseadas em componentes, principalmente por que *mashup* atende a necessidades situacionais e utiliza tecnologias simples e fáceis de serem utilizadas, como por exemplo, REST-*full Web Services* ou RSS/*Atom* feeds. Ele classifica o desenvolvimento *mashup* como manual, onde aplicações *mashup* são desenvolvidas sem o auxílio de ferramentas, utilizando tecnologias de programação *web,* e com o uso de ferramentas, onde o uso de ferramentas específicas simplificam o desenvolvimento, permitindo que usuários finais criem seus próprios *mashups.*

O desenvolvimento *mashup* manual requer conhecimento em programação e nos esquemas e semântica das fontes de dados, bem como na convenção para troca de mensagens. Por exemplo, o desenvolvimento de uma aplicação como o *HousingMaps*[3] envolve acesso a duas fontes de dados distintas (*Craiglist* e *Google Maps*), conhecimento da API *Javascript* do *Google Maps* e processamento da lista de imóveis do *Craiglist* em formato RSS. Essa abordagem torna o desenvolvimento trabalhoso e lento e inviabiliza a criação de *mashup* pelo usuário final.

O desenvolvimento com o uso de ferramentas acelera o processo de desenvolvimento e também permite que os usuários finais criem os seus próprios *mashup* com a ajuda de ferramentas e *frameworks* específicos. Segundo Yu *et al.* [2008] essas ferramentas possuem uma mistura de diversas abordagens de composição.

Segundo Liu *et al*. [2007], desenvolvedores *mashup* enfrentam dificuldades relacionadas a ausência de informações, mapeamento de dados incompletos e a semântica das informações entre os vários conjuntos de dados utilizados pelas aplicações *mashup*. Por exemplo, o registro de oportunidades do cliente poderá ser inserido de forma inconsistente, utilizando abreviações de nomes comuns (como "CustomerOpp" em um sistema de CRM e "Cus Opp" em outro), tornando a verificação automática de correspondência uma tarefa não muito simples.

Existem várias ferramentas para desenvolvimente *mashup*. Dentre as mais destacadas estão as ferramentas Yahoo Pipes, Microsoft Popfly e Intel Mash Maker. A Tabela 1 apresenta as características gerais destas três ferramentas, baseado no estudo apresentado por Lorenzo *et al*. [2009].

Tabela 1- Ferramentas *mashup*

[3] O mashup HousingMaps é um exemplo que combina dados de imóveis da Craiglist com mapas do Google Maps para auxiliar a procura por imóveis (http://www.housingmaps.com/).

Ferramenta	Características gerais
Yahoo Pipes[4]	Ferramenta web onde os usuários podem construir aplicações *mashup* a partir de web *feeds*[5], páginas *web* e outros serviços. Um *pipe* pode ser composto por um ou mais módulos, cada um realizando uma única tarefa como retornar *feeds* de uma fonte na *web*, filtrar, ordenar ou misturar *feeds*. Os dados podem ser consumidos através de RSS, JSON ou visualizados pelo *Yahoo Map*.
Microsoft Popfly[6]	Ferramenta web que permite que os usuários criem *mashup* combinando dados e fontes de mídia. O *mashup* é construído conectando blocos (*blocks*). Cada bloco está associado a um serviço como *Flickr*[7] e expõe uma ou mais funcionalidades. O foco da ferramenta é maior na apresentação das informações do que na manipulação de dados.
Intel Mash Maker[8]	Ferramenta para edição, consulta e manipulação de dados na *web*. Permite a criação de *mashup* combinando conteúdo de diferentes páginas *web*.

Lorenzo *et al*. [2009] apresenta ainda um estudo sobre as principais ferramentas para o desenvolvimento de *mashups,* com foco em integração de dados. O resultado da análise realizada no estudo para as três ferramentas apresentadas na Tabela 1 é apresentado na Tabela 2, a qual foi elaborada baseada no trabalho de Lorenzo *et al*. [2009], atendendo aos objetivos deste trabalho. O "X" na célula indica que a ferramenta atende ao critério. A tabela apresenta os diferentes aspectos das aplicações *mashup* sob o ponto de vista da integração de dados. A seguir é apresentado um resumo dimensões apresentadas no estudo.

- Formato de dados: A dimensão formato de acesso de dados recomenda o uso de API existentes (como destacado por [Murugesan, 2007]) para tratar o problema da integração entre diferentes formatos de dados, como por exemplo, XML, HTML e CSV de fontes heterogêneas, para a troca de informações entre aplicações - características inerentes das aplicações *mashup*. Algumas fontes de dados não expõem seus dados através de API, neste caso deve-se utilizar outra técnica para extração de dados. Normalmente essas fontes de dados, além de estarem em diferentes formatos, também possuem semântica diferente.
- Modelo de dados interno: A dimensão modelo de dados interno corresponde a um esquema global que representa uma visão unificada (integrada) dos dados e podem ser baseados em um modelo XML ou de objetos. Para instanciar um modelo de dados interno a partir de uma fonte de dados externa,

[4] http://pipes.yahoo.com/pipes/

[5] Arquivos *feed* são listas de atualização de conteúdo de um determinado site, escritos com especificações baseadas em XML.

[6] http://*www.popfly.com/*

[7] http://www.flickr.com/

[8] http://mashmaker.intel.com/web/

as ferramentas *mashup* dispõem de estratégias para especificar o mapeamento de dados do modelo externo para o modelo interno.

- Mapeamento de dados: A dimensão mapeamento de dados apresentada por Lorenzo *et al*. [2009] contém três estratégias para mapeamento: manual, semi-automático e automático. No mapeamento manual, todo mapeamento entre o modelo de dados interno e a fonte de dados é especificado manualmente, um a um, pelo desenvolvedor da aplicação. O mapeamento semi-automático utiliza metadados para propor configurações de mapeamentos possíveis, no entanto o usuário deve confirmar as configurações sugeridas e ajustá-las manualmente. Na forma automática, todo mapeamento entre os dois modelos é gerado automaticamente, sem intervenção do usuário. Segundo os autores, o mapeamento automático ainda é um desafio na área de integração de dados, e até o momento não é suportado por nenhuma ferramenta *mashup*.
- Atualização de dados: Visto que os dados sofrem atualizações freqüentes e que essas atualizações devem ser propagadas para os usuários interessados, esta dimensão apresenta duas estratégias *pull e push* [Bhide *et al.*, 2002] que irão tratar a forma com que os dados serão sincronizados entre a fonte de dados e os usuários interessados. Na estratégia *pull*, os usuários (clientes) realizam requisições para o servidor em busca de atualizações, o que pode gerar uma sobrecarga do servidor dependendo da freqüência das requisições. No modelo *push,* ao invés de realizar requisições ao servidor, os usuários se registram no servidor especificando o conteúdo de seu interesse. O servidor transmite os dados para os clientes quando ocorrer modificações no conteúdo. A desvantagem dessa abordagem é que o cliente pode estar ocupado, processando outra tarefa, no momento da transmissão, ocorrendo um atraso no processamento (normalmente utiliza-se processamento assíncrono para minimizar os impactos no processamento). Existe ainda a estratégia *pull* global, onde todas as fontes de dados são atualizadas ao mesmo tempo. Esta estratégia é normalmente encontrada nas ferramentas *mashup*.
- Saída do *mashup*: Esta dimensão trata da exportação do *mashup* para outro formato desejado, para reuso ou composição em outra aplicação. Eles apresentam duas categorias de saída distintas: orientada a humanos e orientada a processos. Na saída orientada a humanos, a visualização é feita através de mapas, páginas HTML e etc. Na saída orientada a processos, o resultado da exportação é utilizado por um processo computacional, especialmente onde a integração de dados necessita de algum tipo de processamento, como, por exemplo, a descoberta de conhecimento em um processo. Pode ocorrer, em algum momento, que essa categoria inclua a primeira, por exemplo, uma saída RSS pode ao mesmo tempo ser visualizada em uma saída HTML como também por outras aplicações para processamento de outras tarefas.
- Extensibilidade: Esta dimensão trata de quão extensíveis são as ferramentas, ou seja, a capacidade da ferramenta em suportar funcionalidades adicionais, normalmente definidas pelo usuário. Existem duas formas de definir e utilizar essas funcionalidades: componentes e dados. As funcionalidades podem estar embutidas na ferramenta, neste caso o código da funcionalidade é adicionado à ferramenta utilizando linguagem de programação específica, ou seja, pela implementação/extensão de componentes. No caso de extensibilidade via dados, as funcionalidades estão externas a ferramenta. Neste caso,

um serviço que oferece a funcionalidade desejada (os serviços podem ser consumidos utilizando tecnologias como REST e SOAP na ferramenta *Yahoo Pipes*).

- Compartilhamento: *Mashups* são baseados em tecnologias emergentes da *web* 2.0, onde os usuários podem criar e compartilhar informações de forma simples e fácil. Neste contexto, a dimensão compartilhamento trata dos aspectos referentes à privacidade e à segurança no compartilhamento de recursos nas aplicações *mashup*. As políticas de compartilhamento podem ser implementadas de diversas formas. Por exemplo, a política total permite acesso de leitura ao código fonte, dados e saída da aplicação *mashup*, enquanto que a política parcial restringe o acesso somente a leitura do código. Os autores ressaltam que a solução dos problemas relacionados à segurança e privacidade das informações ainda é um grande desafio.

Maiores detalhes sobre os critérios avaliados (linhas da tabela) são apresentados por Lorenzo *et al.* [2009].

Tabela 2 - Resultado da análise das ferramentas apresentadas por Lorenzo *et al*. [2009] modificada

	Ferramentas	Yahoo Pipes	Microsoft Popfly	Intel Mash Maker
Formato/Acesso de Dados	Protocolo	REST	REST, SOAP	REST
	Formato Dado	XML, RSS, ATOM, JSON, HTML, CSV, RDF e Imagem	XML, RSS, Imagem e Vídeo	HTML
	Banco de Dados[9]			
Modelo de Dados Interno	XML	X		X
	Objeto		X	
Mapeamento de dados	Manual		X	X
	Semi automático			
	Automático		X	
Atualização de Dados	Pull	X	X	X
	Push			
	Pull Global	X	X	X
Saída Mashup	Orientada para humanos	X		
	Orientada para processos	X	X	X
Extensibilidade	Componentes	X	X	X
	Dados			X
Compartilhamento	Total	X	X	X
	Parcial	X		
	Nenhum	X	X	X
	Somente leitura	X	X	X
	Leitura/Escrita			
	Todos usuários	X	X	X
	Grupos			
	Usuário individual	X	X	X

Segundo Lorenzo *et al*. [2009] as ferramentas *mashup* foram projetadas para tratar dados na *web*, o que pode ser considerado uma vantagem, visto que oferecem suporte aos protocolos SOAP e REST e que os mesmos são utilizados para disponibilizar serviços fornecidos pela maioria das APIs. Por outro lado, também argumentamos que há desvantagens com relação aos dados disponíveis em *desktops*, que não podem ser aces-

[9] As únicas ferramentas que oferecem suporte a banco de dados são Damia [SIMMEN *et al*., 2008] e Apatar (http://www.apatar.com/).

sados da mesma forma, ou seja, não é possível integrar aplicações, como por exemplo, planilhas Excel e outros programas, criados pelos usuários e que não estão na web. Outras desvantagens aparecem em relação às questões de compartilhamento e segurança dos dados, que permanecem como pontos em aberto no desenvolvimento de *mashups*.

Segundo Lorenzo *et al.* [2009], as ferramentas *mashup* exigem algum conhecimento de programação utilizando instruções como *javascript*. As funcionalidades são programadas em um ambiente que oferece uma *interface* gráfica para realizar parte das operações. Até o momento não existem ferramentas que exijam pouco ou nenhum esforço de programação para o desenvolvimento de *mashups*. Esta característica é intrínseca para poder afirmar que as ferramentas são voltadas para o usuário final.

Além das ferramentas caracterizadas na Tabela 1 e Tabela 2, vale ressaltar que existem outras ferramentas como as listadas por Murugesan [2007], o qual apresenta a seguinte relação de ferramentas para o desenvolvimento de *web mashups*: Above All Studio[10], ActionBridge[11], Dapper[12], DataMashups[13], JackBuilder[14], Process Engine[15], Ratchet-X Studio[16], RSSBus[17].

Ele destaca o uso da ferramenta *DataMashups* para a criação de *mashup* através de componentes pré existentes, *feeds*, e acesso a banco de dados local ou remoto.

4. Exemplos de Uso

Nesta seção apresentaremos o *mashup HousingMaps*[18] e três alternativas de desenvolvimento com as ferramentas apresentadas na seção seção anterior.

O *mashup HousingMaps*[19] é um exemplo que combina dados de imóveis da *Craiglist* [20] com mapas do *Google Maps*[21] para auxiliar a procura por imóveis. A *Craiglist* disponibiliza anúncios gratuitos aos usuários, como ofertas de emprego, imóveis, serviços etc. de centenas de cidades ao redor do planeta. O *Google Maps* é um serviço de visualização de mapas e imagens de satélite gratuito da *Google*. A Figura 3 apresenta uma consulta dos imóveis para aluguel em Miami no valor entre 500 e 100 dolares americanos. Os imóveis são apresentados no mapa e ao clicá-los é possível obter fotos e informações como valor e telefone de contato obtidos através da *Craiglist*.

---

[10] http://www.aboveallsoftware.com/product_studio.asp

[11] http://www.nimaya.com/salesforce/index.asp

[12] http://www.dappit.com

[13] http://DataMashups.com

[14] http://jackbe.com/Products/ide.php

[15] http://www.procession.com/products/process_engine.html

[16] http://www.ratchetsoft.com/

[17] http://rssbus.com

[18] http://www.housingmaps.com/

[19] http://www.housingmaps.com/

[20] http://www.craigslist.org/about/sites

[21] http://maps.google.com.br/

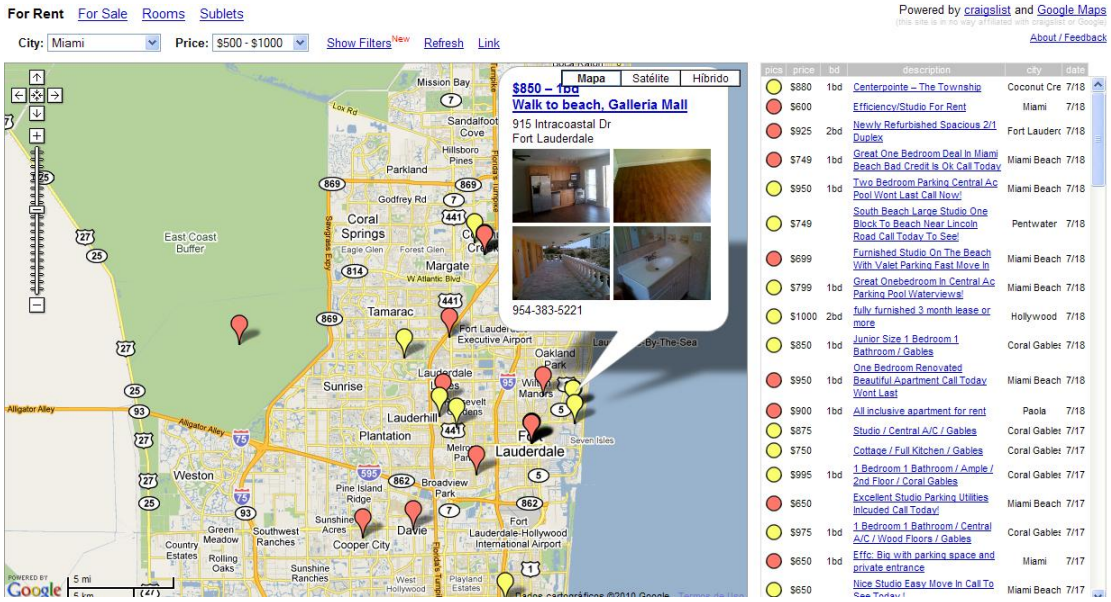

Figura 3 - Consulta de imóveis em Miami utilizando o *HousingMaps*

Yu *et al.* [2008] apresentam um estudo sobre as ferramentas mais populares para *mashups*, sob o ponto de vista do usuário final, e demonstra como elas podem suportar o desenvolvimento de uma aplicação real como a *HousingMaps*. Neste trabalho, serão apresentados alguns detalhes da construção deste aplicativo nas ferramentas *Yahoo Pipes, Microsoft Popfly* e *Intel Mash Maker*.

A construção do *HousingMaps* utilizando o *Yahoo Pipes* não permitiu a implementação da *interface* do usuário, apresentada na Figura 3. Isto por que a ferramenta não provê o desenvolvimento de *interfaces*. Foi utilizado um *pipe* para processar o *feed* RSS da *Craiglist* identificando as informações de localização (*geocodes*). As informações de localização complementaram o *feed* da *Craiglist* com um *link* que permite que os usuários vejam o endereço do imóvel no mapa através da API *Google Maps*. Ou seja, nesta implementação, é apresentada uma lista de imóveis (*feed*) integrada com o *Google Maps*. Quando um imóvel é clicado é apresentada sua posição no mapa, ao invés de todos os imóveis serem apresentado conforme a figura 3.

Para a construção do aplicativo na ferramenta *Microsoft Popfly* foram necessários três blocos (*blocks*). Um bloco para o RSS *feed* da *Craiglist,* um bloco para o mapa e outro para tabela. Foi utilizado o *Virtual Earth*[22] ao invés do *Google Maps*. Os três blocos foram arrastados para a área de desenvolvimento da ferramenta. A saída do bloco RSS contendo o *feed* da *Craiglist* foi conectada aos outros dois blocos, ao do Virtual Earth e ao bloco RSS. Configurações adicionais utilizando *JavaScript* podem ser necessárias.

A ferramenta *Intel Mash Maker* trabalha com conteúdo HTML ao invés de RSS. Para a construção do aplicativo *HousingMaps*, as páginas da *Craiglist* foram anotadas. Em seguida o conteúdo da *Craiglist* foi copiado e colado em páginas do *Google Maps.*

[22] http://www.microsoft.com/maps/

5. Conclusões

Mashup pode ser considerada uma tecnologia emergente. Sua essência está em simplificar a composição de serviços baseado na composição de serviço de SOA, possibilitando a criação de aplicações de forma mais simples e ágil. A idéia central é permitir que usuários finais desenvolvam aplicações resultantes da composição de um ou mais serviços, combinando funcionalidades e recursos de diferentes fontes de dados em uma nova aplicação *web*.

A composição de serviços *mashup* pode ser vista como uma simplificação da composição de serviços em SOA, pois utiliza tecnologias mais simples (*lightweight*) como REST, RSS/ATON e JSON do que as utilizadas normalmente em SOA e são focadas em um universo maior de consumidores, geograficamente distribuídos pela internet. A combinação das informações pode ser realizada através de ferramentas que permitam a criação de aplicativos em uma abordagem Web 2.0. Por outro lado existe um grande problema relacionado à integração de dados e o uso de *mashups*. Alguns autores não recomendam o uso de *mashup* para integração de processos de negócio complexos. O Gartner [2006] alerta que, pelo fato de combinar dados e lógica de diversas fontes de dados, os *mashups* estão sujeitos a falhas em qualquer uma dessas fontes.

Das ferramentas para o desenvolvimento de *mashup*s analisadas nesse trabalho, selecionamos a ferramenta *Intel Mash Maker* para a implementação de um exemplo similando ao *HousingMaps*. A partir de um *feed* de imóveis da *Craiglist* apresentamos as ofertas de imóveis no Google Maps, sem nenhum esforço de programação em um ambiente simples e fácil de usar. Com isso podemos afirmar que existem ferramentas para o desenvolvimento *mashup* voltadas para o usuário final, ou seja, sem nenhum conhecimento de programação. É possível encontrar uma grande quantidade de *mashup* pronto para uso na *web*. Um bom repositório é o *Programmable Web*[23], atualmente com mais de 2000 APIs e 5000 *mashups*.

Um dos aspectos observados durante o estudo é a limitação das soluções apresentadas em endereçar os problemas referentes à segurança nas aplicações *mashup* o que demonstrou ser um ponto em aberto nessa área de pesquisa.

Algumas iniciativas para promover e padronizar a adoção de soluções *mashup* estão sendo conduzidas pelo consórcio OMA[24] (*Open Mashup Alliance*) e com a criação e o desenvolvimento de uma especificação baseada em XML chamada EMML[25] (*Enterprise Markup Mashup Language*) para a criação de *mashups*.

6. Referências Bibliográficas

BOZZON, A., BRAMBILLA, M., FACCA, F.M., CARUGHUI, G.F., 2009. **A Conceptual Modeling Approach to Business Service Mashup Development**. In: IEEE International Conference on Web Services (*ICWS'09*).

BHIDE, M., DEOLASEE, P., KATKAR, A., PANCHBUDHE, A., RAMAMRITHAM, K., SHENOY, P., 2002. **Adaptive push-pull: Disseminating dynamic web data**. In: IEEE Transactions on Computers.

[23] http://www.programmableweb.com/

[24] http://www.openmashup.org/

[25] http://www.openmashup.org/omadocs/v1.0/index.html

BPEL, Business Process Execution Language, disponível em <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel>, 2008.

BPMN, Business Process Model and Notation, disponível em <http://www.omg.org/spec/BPMN/2.0/Beta2/PDF/>, 2010.

ERL, T., 2005. **Service-Oriented Architecture: concepts, technology, and design,** Prentice Hall.

GARTNER, 2006. Emerging Technologies Hype Cycle Highlights Key Technology Themes. Disponível em http://www.gartner.com/it/page.jsp?id=495475. Acessado em 11/07/2010.

HEWITT, E. **Java SOA Cookbook**, O'Reilly, 2009.

HOYER, V., **STANOEVSKA-SLABEVA**, K., **JANNER**, T., **SCHROTH**, C., 2008. Enterprise Mashups: Design Principles towards the Long Tail of User Needs. In: IEEE International Conference on Services Computing (*SCC'08*).

JHINGRAN, A., 2006. **Enterprise Information Mashups: Integrating Information Simply**. In: The 32nd International Conference on Very Large Databases.

JOSUTTIS, N., 2007**.** ***SOA*** **in Pratice: The Art of Distributed System Design**. O'Reilly, CA.

LIU, X., HUI, Y., SUN, W., LIANG, H., 2007. **Towards Service Composition Based on Mashup**. In: IEEE Service Computing, pp. 332–339.

LIU, Y., LIANG, X., XU, L., STAPLES, M., ZHU, L., 2009. **Using Architecture Integration Patterns to Compose Enterprise Mashups.** In: WICSA/ECSA.

LORENZO, G. D., HACID, H., PAIK, H., BENATALLAH, B. 2009. **Data integration in mashups**. In: SIGMOD.

MAXIMILIEN, E. M., RANABAHU, A., TAI, S., 2007. **Swashup: situational Web applications Mashups**. In: Conference on Object-oriented programming systems and applications (OOPSLA).

MERRIL, D., 2009. **Mashups: The new breed of Web app. An introduction to mashup**. Disponível em http://www.ibm.com/developerworks/xml/library/x-mashups.html. Acessado em 18 Jul. 2010

MURUGESAN, S., 2007. Understanding web 2.0 .Disponível em http://www.computer.org/portal/web/buildyourcareer/fa009.

PEDRAZA, G., ESTUBLIER, J., 2009. **Distributed Orchestration Versus Choreography: The FOCAS Approach.** In: International Conference on Software Process (*ICSP'09*).

PAPAZOGLOU, M. P., TRAVERSO, P., DUTSDAR, S., LEYMANN, F., 2007. **Service-**

Oriented Computing: State of the Art and Research Challenges. In: IEEE

Computer Society, vol. 40, issue 11, pp. 38-45.

SIMMEN, D.E., ALTINEL, M., MARKL, V., PADMANABHAN, S., SINGH, A., 2008. **Damia: data mashups for intranet applications**. In: SIGMOD '08, pages 1171–1182.

TUCHINDA, R., SZEKELY, P., KNOBLOCK, C.A., 2008. **Building Mashups By Example**. In: 13th international conference on Intelligent user interfaces.

WATTS, S., 2001. **Mashups – The evolution of the SOA, Part 2: Situational applications and the Mashup Ecosystem**. Disponível em

http://www.ibm.com/developerworks/webservices/library/ws-soa-mashups2/. Acessado em 12 Jun. 2010

WSCI, Web Service Choreography Language, disponível em <http://www.w3.org/TR/wsci/>, 2002.

YU, J., BENATALLAH, B., CASATI, F., DANIEL, F., 2008. **Understanding Mashup Development**. In: Internet Computing, pp. 44-52

ZOU, J., PLAVLOVSKI, C.J., 2007. **Towards Accountable Enterprise Mashup Services**. In: IEEE International Conference on e-Business Engineering.