Uma Abordagem para a Percepção através da Orientação a Aspectos em uma Infra-estrutura de *Groupware*

Michel Ridwan Oei[1], Rita Suzana P. Maciel[1,2], José Maria N. David[1], Adriano Augusto de Oliveira Bastos[1], Leandro de Oliveira Menezes[1]

[1]Faculdade Ruy Barbosa – Salvador – BA

[2]Universidade do Estado da Bahia (UNEB) – Salvador – BA

{michel, rsuzana, josemaria, abastos, leomenezes}@frb.br

Resumo. *Soluções para apoiar a percepção em ambientes de groupware têm sido propostas. Tais soluções freqüentemente representam interesses transversais relacionados à percepção através da orientação a objetos, resultando em componentes fortemente acoplados , bem como na redundância e na dispersão das funcionalidades em diversas partes do sistema. O objetivo deste artigo é apresentar os experimentos realizados com um serviço de percepção, denominado Aw2SOA (Awareness to Service-Oriented Architecture), implementado de acordo com os princípios de SOA (Service-Oriented Architecture) e os conceitos de programação orientada a aspectos. Esses experimentos foram conduzidos através da integração deste serviço à WGWSOA (Web-based Groupware Service-Oriented Architecture).*

Abstract. *Solutions have been proposed aimed to support awareness in groupware. These solutions usually represent crosscut concerns related to awareness by the use of object-oriented programming. As a result, tightly-coupled components have been generated as well as code redundancy and scattering. This work aims to present experiments results carried out with an awareness service named Aw2SOA (Awareness to Service Oriented Architecture) It was implemented according with SOA (Service Oriented Architecture) principles and AOP (Aspect Oriented Programming). These experiments have been carried out in order to evaluate this solution in a WGWSOA (Web-based Groupware Service Oriented Architecture) environment.*

1. Introdução

Atividades complexas que antes eram realizadas individualmente passaram a ser conduzidas colaborativamente em grupos no intuito de atender à demanda por eficiência, produtividade e qualidade. Fazer com que estes grupos colaborem não é uma tarefa fácil, e um dos problemas que dificultam a colaboração é a falta de contexto entre os participantes do grupo. A percepção em *groupware* consiste no conhecimento das atividades do grupo com a intenção de prover contexto para os participantes. Quando analisada numa granularidade fina, a percepção pode ser definida como um evento (*event-based awareness*) que deve ser distribuído a todos os participantes interessados. [Dourish e Bellotti 1992]. Eventos como a troca de mensagem entre usuários, a

conclusão de uma tarefa ou a edição de um artefato compartilhado estão presentes em diversas atividades apoiadas por um *groupware*. Tais eventos são utilizados para apoiar a comunicação, a coordenação e a cooperação. Funcionalidades relacionadas a tais eventos são muitas vezes identificadas em diferentes módulos (por exemplo, componentes, objetos e serviços), cada um com diferentes níveis de abstração.

Mesmo que alguns requisitos não-funcionais sejam considerados (por exemplo, portabilidade, escalabilidade e interoperabilidade), é possível encontrar funcionalidades similares implementadas em diferentes módulos. Tais funcionalidades estão relacionadas sobretudo ao requisito funcional "fornecer percepção". Como resultado, atividades de manutenção e evolução de requisitos tornam-se difíceis de serem apoiadas.

A representação de interesses transversais através do paradigma da orientação a objetos tem resultado em um código fortemente acoplado, bem como o espalhamento de funcionalidades em diferentes partes da aplicação. Com o intuito de resolver tais problemas , bem como modularizar interesses transversais e facilitar tanto a manutenção quanto a compreensão do código, Programação Orientada a Aspectos [Kiczales *et al.* 1997], [Kiczales *et al.* 2001] tem sido proposta.

A WGWSOA (*Web-based Groupware Service-Oriented Architecture*) é uma arquitetura orientada a serviços de *middleware* cujo objetivo é apoiar a reutilização e a interoperabilidade entre sistemas colaborativos [Maciel e David 2007]. É uma arquitetura na qual os serviços utilizados são "serviços de *middleware*" de acordo com Schantz e Schmidt (2001). Nesta arquitetura, serviços específicos de coordenação, cooperação, comunicação e percepção foram implementados. Eles apóiam o projeto e a implementação de *groupware* através da composição e da reutilização, entre outros princípios de SOA [Erl 2005]. Entretanto, o suporte à percepção ainda não foi adequadamente apoiado pela orientação a objetos, considerando sobretudo os requisitos previamente mencionados.

Este artigo apresenta um serviço de percepção denominado Aw2SOA [Bastos *et al.* 2008], desenvolvido para apoiar a percepção sobre ações ocorridas tanto nas aplicações quanto nos serviços da WGWSOA. Ao mesmo tempo as características de sua arquitetura, implementação e testes são detalhadas. Em Bastos *et al.* (2008) a proposta foi especificada e apresentada, porém as avaliações da solução não foram suficientemente exploradas. A Seção 2 descreve a arquitetura da WGWSOA. A solução proposta é discutida na Seção 3 bem como as avaliações do serviço. Na Seção 4 são apresentadas abordagens para percepção em *groupware*. Por fim, na Seção 5 as conclusões do artigo são apresentadas.

2. WGWSOA: uma infra-estrutura para apoiar o desenvolvimento de *groupware*

Sistemas colaborativos distribuídos são aplicações que possuem requisitos complexos difíceis de serem implementados nas atuais arquiteturas de *groupware* [Maciel e David 2007]. Esta complexidade está relacionada com os diversos requisitos de ambientes distintos, os quais necessitam socializar informações relacionadas com a semântica das atividades, bem como as informações comuns produzidas pelas aplicações de *groupware* utilizadas. Além disso, existem algumas limitações no que se refere à evolução das funcionalidades do *groupware* em ambiente *web*.

Aplicações de *groupware* distribuídas são normalmente construídas através do uso de plataformas *middleware* [Maciel e David 2007]. Estas formam uma camada de software residente entre a aplicação e o sistema operacional, sua função é ocultar a heterogeneidade entre plataformas de hardware e software, bem como melhorar a transparência entre sistemas distribuídos através do uso de serviços [Schantz e Schmidt 2001]. Tendo em vista a alta complexidade para manter a interoperabilidade entre aplicações de *groupware*, mesmo que construídas sobre uma mesma plataforma de *middleware* (por exemplo, J2EE, CORBA e .NET), algumas soluções têm sido propostas baseada na Arquitetura Orientada a Serviços (SOA).

SOA pode ser entendida como um paradigma arquitetural que viabiliza a criação de serviços de negócio com baixo acoplamento e interoperáveis entre si, os quais podem ser facilmente compartilhados [Erl 2005]. Através de seu uso possibilita-se um fraco acoplamento, pois o consumidor não tem acesso a variáveis e métodos internos da implementação do serviço, apoiando outros princípios, como reutilização, composição e autonomia . Assim, a criação e o uso de *middleware*, e as arquiteturas neles baseadas, têm sido motivados pelas exigências dos novos ciclos de desenvolvimento de software, os quais necessitam de maior agilidade, menor esforço e maior reuso [Schantz e Schmidt 2001].

A WGWSOA é uma arquitetura baseada em serviços de *middleware* que apóiam aplicações de *groupware*, fornecendo uma camada de abstração com serviços que atendem ao domínio dos sistemas colaborativos. Além disso, promove interoperabilidade entre aplicações de uma mesma, ou de diferentes, plataformas atendendo a requisitos como composição, escalabilidade, facilidade de extensão e baixo acoplamento [Maciel e David 2007].

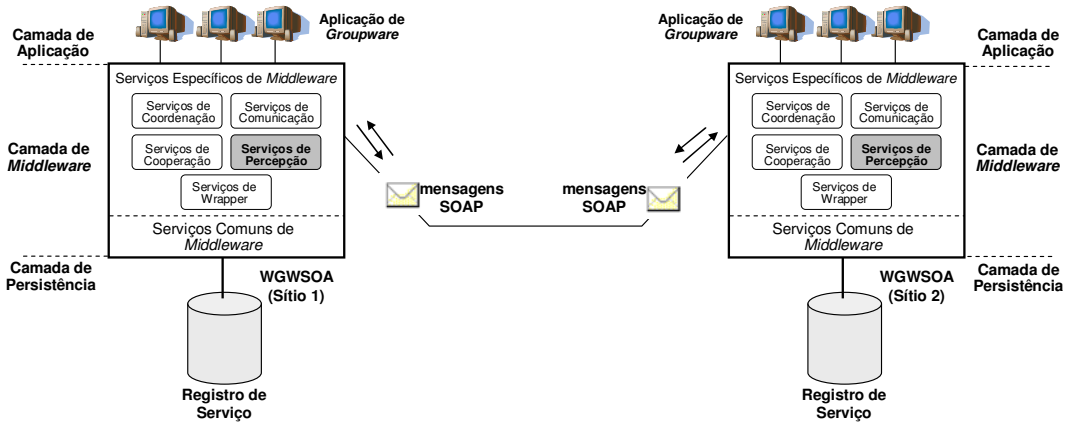

Figura 1. Arquitetura geral da WGWSOA (adaptado de [Maciel e David 2007]).

Conforme ilustrado na Figura 1, a arquitetura da WGWSOA é dividida em três camadas: aplicação, *middleware* e persistência. Na primeira camada, aplicações de *groupware* desenvolvidas utilizando os serviços da camada de *middleware* são encontradas. A camada de *middleware* foi implementada de acordo com a proposta de Schantz e Schmidt (2001), na qual o *middleware* é dividido em serviços específicos e serviços comuns. A última camada está relacionada com o armazenamento e recuperação dos dados referentes aos elementos do *groupware*, tornando transparente aos serviços do *middleware* toda a parte de acesso ao banco de dados.

Ainda ilustrado na Figura 1, a interoperabilidade entre *groupware* distintos pode ser alcançada instanciando em cada sítio uma WGWSOA. Para tornar isso possível, os serviços implementados nesta arquitetura são descritos através do protocolo SOAP (*Simple Object Access Protocol*) e WSDL (*Web Services Description Language*), possibilitando assim aos desenvolvedores tirarem maior proveito na utilização do primeiro para a comunicação através de diferentes implementações em um mesmo *middleware*, e do segundo para proporcionar a interoperabilidade entre diferentes plataformas de *middleware* que utilizam a tecnologia *Web Services*.

Nesta arquitetura os serviços específicos são baseados no modelo de colaboração 3C definido por Ellis, Gibbs e Rein (1991). Eles possuem funcionalidades específicas do domínio da aplicação, por exemplo, sistemas colaborativos (fóruns, chats, lista de discussão). Já os serviços comuns fornecem funcionalidades de alto nível independentes do domínio da aplicação (controle de concorrência, localização de objetos, administração das transações). Esses serviços podem ser compostos em outros serviços ou estendidos para apoiar diferentes modos de colaboração, de modo a atender aos requisitos do *groupware*.

3. Aw2SOA

A Aw2SOA (*Awareness to Service-Oriented Architecture*) é um serviço específico de *middleware*, e tem o propósito de apoiar a percepção sobre ações ocorridas tanto nas aplicações quanto nos serviços da WGWSOA, com o objetivo de propagar informações tais como: o que, onde, quem e quando as ações ocorreram [Bastos *et al.* 2008].

O processo adotado pelo serviço Aw2SOA promove a troca de informações na forma de eventos fornecendo suporte às ações ocorridas tanto nas aplicações como nos serviços que estão contidos no *middleware*. Estes eventos são notificados no momento de sua ocorrência e armazenados em um repositório podendo ser consultados posteriormente.

3.1 Especificação do serviço e do modelo conceitual

Serviços e aplicações que utilizam a Aw2SOA são classificados como atores-produtores quando geram eventos, ou atores-consumidores quando estão interessados em manipular as informações dos eventos. De modo a assegurar um baixo acoplamento com o serviço de percepção, e administrar a relação de consumo e geração, um perfil é criado para cada ator. Tais perfis guardam as informações de quais eventos são gerados e consumidos pelo proprietário deste perfil, possibilitando assim a notificação de eventos aos respectivos interessados.

A Figura 2 ilustra as principais funcionalidades do Aw2SOA com um diagrama de casos de uso. Os principais casos de uso da Aw2SOA são: (i) manter perfis para agrupar as informações que relacionam os produtores e consumidores aos eventos; (ii) notificar eventos, através da verificação dos perfis dos consumidores e enviar-lhes eventos de seu interesse; (iii) consultar eventos, prover um meio para busca de eventos ocorridos no passado; (iv) captar evento, interceptar a execução dos métodos dos serviços que representam eventos; (v) receber evento, consiste em receber as informações relacionadas à ocorrência de um evento; (vi) persistir evento, verifica o contexto do evento com o que está cadastrado no perfil e realiza sua persistência.

Atores, que interagem com o serviço, podem ser tanto uma aplicação colaborativa, quanto outro serviço da WGWSOA.

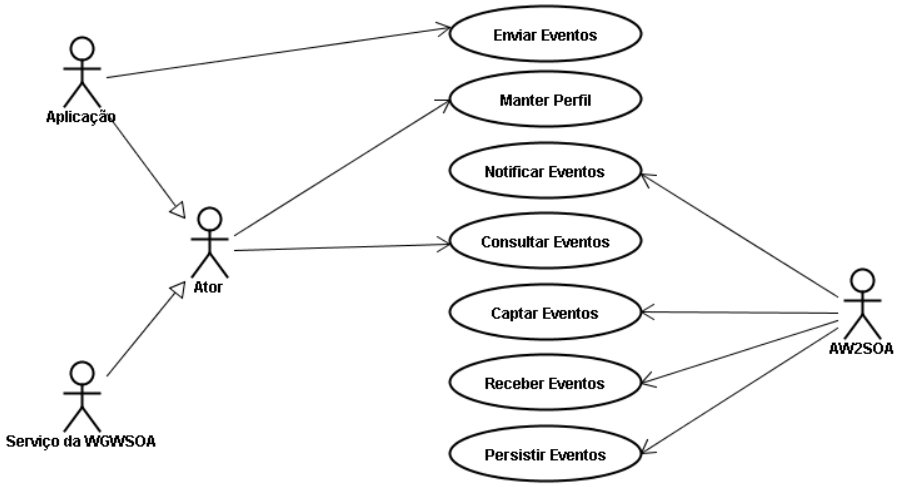

Figura 2. Diagrama de Casos de Uso da Aw2SOA.

A Figura 3 apresenta o modelo conceitual da Aw2SOA para a produção e geração de evento. Cada evento é representado por um nome que o classifica, o nome do ator que o gerou, a hora da ocorrência e o contexto em que ele ocorreu. `Actor` representa as entidades que geram ou consomem os eventos, possuindo os atributos nome e porta que a identifica. Tais entidades podem ser classificadas como aplicação ou serviço.

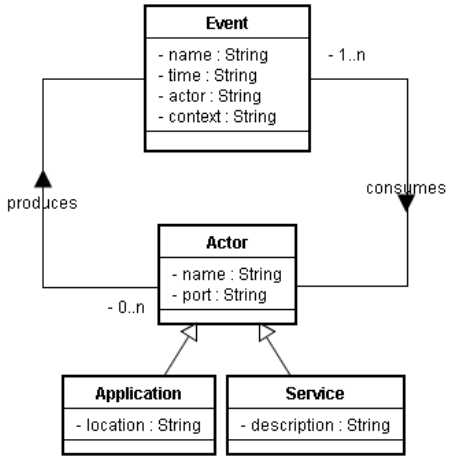

Figura 3. Modelo Conceitual da Aw2SOA.

A implementação da AW2SOA está baseada neste modelo, no qual o principal elemento é o evento através do qual o suporte à percepção é oferecido na WGWSOA.

3.2 Arquitetura

Aw2SOA é composta pelos seguintes serviços: Evento, Notificação, Consulta, Persistência, Gerenciador de Perfil e o Sensor. Seguindo o padrão de projeto `Facade` é

disponibilizada em uma única interface para a utilização destes serviços. A Figura 4 ilustra a arquitetura com estes serviços e suas relações.

O Serviço de Eventos funciona como forma de entrada dos eventos no ambiente de *groupware* no qual a WGWSOA está atuando. Sua interface define os atributos necessários para a criação e contextualização do evento, tais como: o nome do evento e do ator responsável, uma lista de nomes que identificam os parâmetros do evento e outra lista com os valores correspondentes aos parâmetros. Através destes atributos o serviço transforma as informações em um arquivo XML (eXtensible Markup Language), armazenado temporariamente em um buffer de eventos recentes, para fins de notificação e, por fim, é persistido na base de dados para que possa ser consultado posteriormente.

As informações enviadas ao Serviço de Eventos são validadas de acordo com o perfil previamente cadastrado. Para tanto, é feita uma comparação entre os dados recebidos com aqueles contidos na base de dados da instância da WGWSOA garantindo a confiabilidade na estrutura do modelo XML gerada.

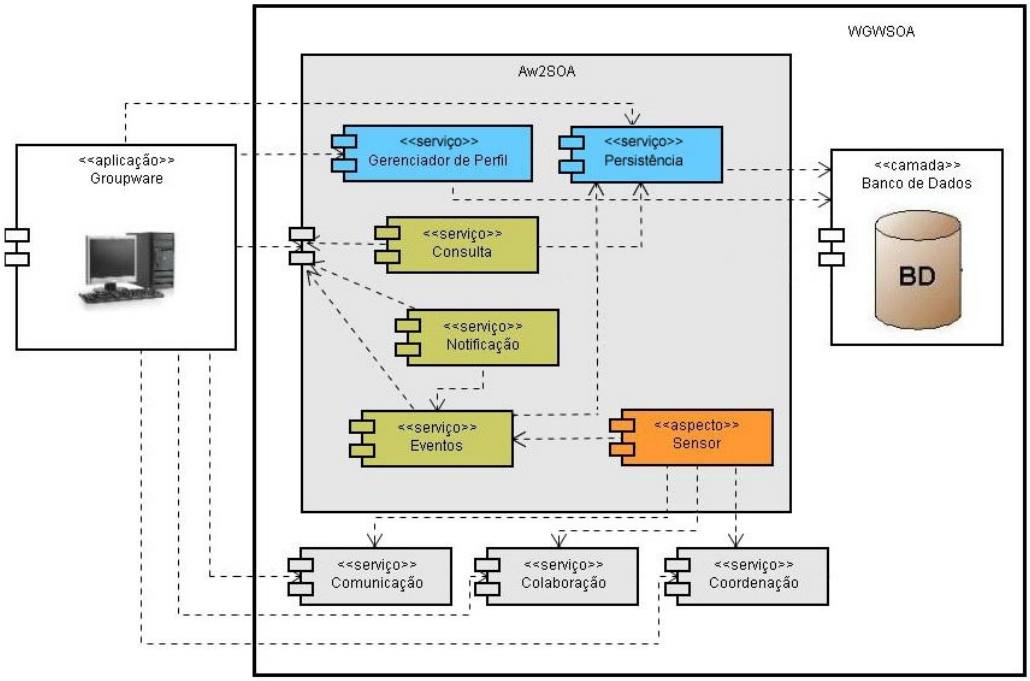

Figura 4. Arquitetura da Aw2SOA.

O Serviço de Notificação é responsável por identificar e agrupar os eventos recentes de interesse das aplicações ou serviços da WGWSOA para que possam ser consumidos. A forma de notificação utilizada segue o modelo *pull*, o qual obriga o consumidor a requisitar os eventos de seu interesse para que eles sejam retornados. A interface do Serviço de Notificação, então, utiliza o nome do perfil como parâmetro para buscar os eventos recentes armazenados no buffer. Após a busca, é armazenada em outro buffer a última requisição realizada pelo consumidor, que será utilizada para evitar o envio de um mesmo evento mais de uma vez, garantindo a integridade do serviço.

O Serviço de Consulta define uma forma das aplicações e serviços obterem informações dos eventos ocorridos no passado. Para a realização da consulta, apenas devem ser informados os dados que determinarão quais os eventos devem ser buscados. Estes dados podem estar relacionados ao nome do ator responsável, ao nome do evento e a um intervalo de ocorrência. Os eventos são buscados na base de dados e então agrupados em um documento XML.

O Sensor é responsável por capturar eventos que ocorreram nos serviços específicos da WGWSOA, foi implementado de forma que seu funcionamento seja independente e transparente em relação aos serviços. Sua função é separar o interesse transversal da percepção e gerar eventos dos serviços. O paradigma orientado a aspectos foi a estratégia adotada para construí-lo, permitindo sua utilização de uma forma não intrusiva.

Enquanto os serviços de consulta, notificação e o sensor são essenciais para o Aw2SOA, os serviços de persistência e gerenciamento de perfil servem de apoio aos demais. O serviço de persistência tem como objetivo abstrair o acesso ao banco de dados e manipular arquivos XML, retirando dos demais serviços esta responsabilidade. Já o gerenciamento de perfil têm a responsabilidade de manter o perfil, proporcionando uma forma de criar, modificar e apagar suas informações.

As tecnologias utilizadas na implementação desta arquitetura foram baseadas na plataforma J2EE, utilizando a especificação DOM (*Document Object Model*) para a manipulação de XML, JPA (*Java Persistence API*) na camada de persistência dos dados, AspectJ na implementação do sensor e a abordagem RMI (*Remote Method Invocation*) para todos os serviços. Para a comunicação entre instâncias distintas da WGWSOA na *web* utilizou-se o protocolo SOAP.

3.3 Implementação do serviço

A tentativa de representar os interesses transversais no paradigma orientado a objetos resulta em forte acoplamento, redundância e dispersão das funcionalidades em diversas partes do sistema [Kiczales *et al.* 2001]. Como forma de modularizar essa transversalidade, facilitando sua manutenção e evolução, utilizamos como estratégia o paradigma orientado a aspectos na implementação.

A POA é comumente utilizada no tratamento de requisitos não funcionais (portabilidade, segurança, escalabilidade etc.) pela característica de estarem presente em diferentes partes do sistema, porém, o requisito percepção, apesar de ser funcional, também está entrelaçado às demais funcionalidades do *groupware*.

A WGWSOA é um *middleware* em evolução, em virtude disso, novos serviços serão implementados, não sendo possível identificar previamente os métodos que serão capturados (*join points*) pelo Sensor. Como forma de oferecer uma solução para este problema e tornar a definição do método a ser capturado de responsabilidade do conhecedor do domínio, foi utilizado o recurso de *Java Annotations*, juntamente com o aspecto. *Java Annotations* são metadados que podem acrescentar informações ao código sem que haja dependência ou modificação na lógica do mesmo.

```
1 public aspect AspectSensor {
2     pointcut interceptEvent():execution(public * br.frb.wgwsoa.services.*.*(..));
3     before():interceptEvent(){
4         try {
5             String methodName = thisJoinPoint.getSignature().getName();
6             Class classType = thisJoinPoint.getSignature().getDeclaringType();
7             Method method = this.getMethod(classType, methodName);
8             System.out.println("AspectSensor.before()111");
9             if (method.isAnnotationPresent(Awareness.class)){
10                System.out.println("AspectSensor.before()222");
11                this.sendEvent(thisJoinPoint, method);
12            }
13        }
14        catch(NoSuchMethodException e){
15            e.printStackTrace();
16        }
17    }
```

Figura 5. AspectSensor na Aw2SOA

Para que eventos sejam gerados a partir dos serviços da WGWSOA, eles devem pertencer ao pacote *services* e seus métodos devem conter a *annotation* `@Awareness`. Devido a essa abordagem, é necessário que seja realizado o *weaving* e o código seja compilado, porém não há necessidade de nenhuma modificação na lógica de negócio para acréscimo da percepção. Desta forma, a manutenção da percepção torna-se transparente, e não interfere na evolução das funcionalidades do serviço da WGWSOA. A Figura 5 apresenta o `AspectSensor` que captura a ocorrência dos métodos através do *pointcut* `interceptEvent` (linha 2), no seu *advice before* (linha 3) são filtrados os métodos que possuem *annotation* `@Awareness`. As informações necessárias para geração do evento são obtidas através de reflexão e encaminhadas para o Serviço de Eventos (linha 10).

Cada evento possui um contexto diferente de acordo com a origem em que foi gerado, por exemplo: (i) qual atividade foi delegada através do serviço de coordenação?; (ii) quem recebeu a mensagem do serviço de comunicação? ; (iii) qual o nome do artefato que foi modificado pelo serviço de cooperação? Para representar os diversos contextos, o evento é codificado através da linguagem *XML,* tornando-o mais flexível, bem como deixando a cargo do responsável pela criação do evento a definição das informações adicionais que devem estar contidas nele, além daquelas que são obrigatórias. A Figura 6 ilustra um modelo de representação do evento em *XML.*

```
1 <?xml version="1.0" encoding="UTF-8"?>
2  <event name="Nome do Evento" responsable="Nome do Ator" timeOccurence="Data e hora">
3         <context>
4            <parametro1 value="VALOR"/>
5            <parametro2 value="VALOR"/>
6            <parametro3 value="VALOR"/>
7                        .
8                        .
9                        .
10           <parametroN value="VALOR"/>
11       </context>
12 </event>
```

Figura 6. Modelo de representação do evento.

A primeira linha contém versão e codificação referentes ao *XML.* Na segunda linha, o nó "*event*" é comum a todos os eventos, e contém os seguintes atributos básicos: o nome do evento, o seu responsável e a hora de ocorrência. As linhas de 4 a 10, referem-se aos nomes dos parâmetros e seus valores, representando a descrição do contexto no qual ocorre o evento.

3.4 Avaliação

Experiências de uso foram realizadas em cenários distintos buscando-se refletir situações de desenvolvimento e uso de *groupware*. Nestas experiências foram observadas as seguintes situações: (1) se o modelo de eventos da Aw2SOA pode ser utilizado nos diversos serviços da WGWSOA e; (2) se a modularização oferecida pelo Aw2SOA para a percepção pode reduzir o impacto em relação às modificações necessárias nos demais serviços.

A primeira questão tem como objetivo verificar a forma de definição dos eventos, desde a criação do perfil até o tratamento destes, nos diferentes contextos de cada domínio. O desenvolvedor do *groupware* pode registrar o seu perfil, enviar eventos para a instância da WGWSOA, e realizar consultas sobre estes ou outros eventos de seu interesse, bem como avaliar o nível de reuso, de flexibilidade e de abstração oferecidos pela Aw2SOA. A segunda questão objetiva identificar o quanto foi possível encapsular a percepção através do paradigma orientado a aspectos, mantendo a lógica de negócio dos demais serviços inalterada.

Para tanto, dois cenários foram construídos. O primeiro teve como objetivo incluir suporte à percepção em quatro serviços de coordenação da WGWSOA. Os serviços escolhidos foram aqueles desenvolvidos para a medição dos seguintes elementos: conflitos ("Conflictmeter", `Conflictmeter.setConflicts`), participação ("Participameter", `Participameter.setParticipations`), contribuição ("Contributiometer", `Contributiometer.setContributions`) e impactos ("Impactmeter", `Impactmeter.setImpacts`) [Toledo *et al.* 2008]. A *annotation* @Awareness foi inserida conforme descrito anteriormente sendo que os métodos escolhidos possuem parâmetros diferentes, logo contextos distintos para cada evento.

A utilização dos eventos retirou dos serviços a responsabilidade de manter uma estrutura de armazenamento, atualização e recuperação de informações que deveriam ser propagadas para as aplicações. Em uma comparação realizada entre os códigos dos serviços, antes e após a composição com o Aw2SOA, foi possível observar uma redução de seiscentas linhas de código.

Em um segundo cenário, os experimentos foram realizados em um *Chat* de uma aplicação para gerência de autoria colaborativa. Os principais requisitos do *Chat* são: permitir a troca e persistência de mensagens entre usuários de uma mesma sala, onde uma sala é definida a partir de seu nome, e exibir a lista de usuários que estão on-line em cada sala.

O primeiro passo foi definir o evento que seria gerado e consumido pelo *Chat*, com o nome `newMessage`, contendo em seu contexto os seguintes valores: `sender`, usuário que enviou a mensagem, `room`, sala em que o usuário está presente e `text`, texto da mensagem. Para compor o *Chat*, além da *Aw2SOA*, foram utilizados mais dois serviços da *WGWSOA,* o serviço de lista de usuários (`UserList`) e de autenticação (`UserLogin`), com o objetivo auxiliar no requisito de exibição dos usuários *on-line*. Para este cenário, foram criados três perfis junto a *Aw2SOA*: um para o *Chat*, que define o `newMessage` como evento gerado e consumido, outro para o `UserLogin`, que define a geração dos eventos `login` e `logoff`, e outro o `UserList`, que consome estes eventos do serviço `UserLogin`.

Após a composição do serviço do *Chat* com Aw2SOA, os desenvolvedores da aplicação perceberam a necessidade de acrescentar mais funcionalidades através dos modelos de eventos consumidos e do serviço de consulta. Sendo assim, foram implementadas outras funcionalidades, tais como: as exibições tanto das conversas anteriores da sala, como dos usuários, através da notificação da hora em que as mensagens foram enviadas e quando os usuários ficaram *on-line*, bem como o *feedback* sobre o estado dos mesmos, por exemplo, "ocioso" ou "escrevendo".

Após a realização dos experimentos é possível afirmar que o grau de abstração dos serviços da *Aw2SOA*, a partir da avaliação das interfaces dos serviços, pode ser classificado como "Detalhado" [Erl 2005], por possuir regras bastante explícitas sobre a lógica de definição dos perfis e a definição do tipo de evento retornado

Durante todos os experimentos, não houve a necessidade de alteração em nenhum dos serviços envolvidos. A abordagem orientada a aspectos utilizada para separação do interesse percepção, cumpriu seu objetivo de retirar da lógica dos demais serviços a necessidade de implementar tal requisito.

4. Abordagens para percepção em *groupware*

Diversas soluções têm sido propo*stas para* tratar a percepção em aplicações de *groupware* distribuídas. Algumas delas util*izam tecn*ologias que suportam o desenvolvimento baseado em componentes, outras foram concebidas como *plug-ins* para serem adaptadas ao *groupware, porém* sem uma abordagem adequad*a para a* interoperabilidade entre plataformas para apoiar a colaboração. Nesta seção, algumas das soluções serão descritas com o objetivo de identificar as suas principais características.

BigWatcher [Pinheiro *et al.* 03] é um *framework* que monitora e contextualiza o trabalho dos participantes de um grupo. Seu principal objetivo é promover a percepção através de eventos ocorridos no passado, que tanto podem ser persistidos quanto apresentados de uma maneira bem flexível. Por ser um *framework*, o mecanismo não implementa todas as funcionalidades necessárias, demandando que extensões sejam realizadas em cada aplicação que deseje utilizá-lo.

NESSIE [Prinz 99] é um ambiente concebido para apoiar a percepção, oferecendo uma infra-estrutura genérica para cada aplicação, juntamente com sensores e indicadores, com o objetivo de notificar eventos. Por meio de um servidor, onde estão configurados os indicadores de consumo do evento, as informações sobre atividades colaborativas são captadas automaticamente por sensores, que em geral, estão associados com os atores e objetos compartilhados das atividades. Eventos em NESSIE, não necessariamente, são gerados automaticamente por sensores associados a um trabalho específico, eles podem ser gerados quando ocorre uma mudança do cliente, ou quando são enviados diretamente para o Servidor NESSIE através do HTTP e de *scripts* CGI.

InterDoc [Maciel *et al.* 05] é um ambiente de suporte à interoperabilidade entre ferramentas de autoria colaborativa de documentos, o seu serviço de percepção utiliza filtros para captar dados da aplicação e enviá-los para outros servidores criando uma camada intermediária entre o fluxo de dados [Almeida *et al.* 07]. Apesar de ser transparente para a aplicação o servidor possui toda a implementação definida para

apoiar o InterDoc (Usuários, Atividades e Planejamento), por este motivo, toda modificação deve ser acrescentada aos novos servidores.

No serviço Aw2SOA, a percepção é considerada como um interesse transversal e que deve ser implementada independente da aplicação. Usando o conceito de POA, os eventos são capturados a partir de outros serviços e podem ser gerados de maneira mais simples pela aplicação, sendo necessário apenas enviar as informações do evento através de um método disponibilizado pelo serviço. Para o processo de notificação, basta então que o serviço ou aplicação sigam um protocolo, não sendo necessário implementar extensões para utilizar o serviço.

Aw2SOA pode ser diferenciada das demais abordagens por permitir o acréscimo da percepção de maneira flexível, possibilitando que os eventos sejam eleitos durante a fase de projeto do *groupware,* sem a necessidade de compreensão de uma API, de uma integração com outro sistema ou de inserção de código na regra de negócio dos serviços.

5. Conclusões

A abordagem orientada a serviços é sugerida na literatura como forma de promover a reutilização. Implementado como um serviço da WGWSOA, o Aw2SOA cumpriu este requisito nas diversas situações em que foi usado. O objetivo principal deste trabalho foi especificar, implementar e avaliar um serviço capaz de tratar as características de percepção dos serviços específicos contidos na WGWSOA. Para torná-lo reutilizável e com baixo acoplamento*,* conceitos de SOA foram aplicados à sua arquitetura, tornando a *Aw2SOA* resultado da composição de outros serviços.

O paradigma orientado a aspectos foi a estratégia adotada para separar os interesses transversais e a geração de eventos dos serviços. A tentativa de representar tais interesses no paradigma orientado a objetos resulta em forte acoplamento, redundância e dispersão das funcionalidades em diversas partes do sistema. Portanto, como forma de modularizar essa transversalidade, facilitando sua manutenção e evolução, utilizou-se o paradigma orientado a aspectos.

A solução proposta foi utilizada em experimentos partir da integração de novos serviços da WGWSOA e do desenvolvimento de aplicações de *groupware*. Através dos experimentos foi possível evidenciar que a estratégia de prover a percepção na camada de serviços de uma arquitetura orientada a serviços , através da abordagem da orientação a aspectos, pode manter os níveis de reuso e acoplamento fraco definidos para a mesma .

Como trabalhos futuros, outros experimentos devem ser conduzidos no sentido de revelar novas questões em relação à abordagem utilizada, como por exemplo o seu efeito em outros requisitos não-funcionais. A WGWSOA está sendo implementada em uma plataforma distinta e heterogênea em relação à instância atual, sendo assim o Aw2SOA deve ser também avaliado neste novo contexto.

6. Referências

Almeida, L., Maciel, R. S., David, J. M. N. (2007) “Uma Proposta para Serviços de Percepção no Ambiente Interdoc”. Revista CienteFico.com, v. I, p. 91-100.

Bastos, A., Oei, M., Menezes, L., Pitangueira, R. S., David, J. M. N. (2008) "Aw2SOA: An Aspect-Oriented Awareness Service for Distributed Groupware". In: Proc. of the 2008 12th International Conference on Computer Supported Cooperative Work in Design, p. 404-409, v. I, Xi´an, China.

Dourish, P., Bellotti, V. (1992) "Awareness and Coordination in Shared Workspaces", In: Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW'92), p.107-114, Toronto, Ontario.

Ellis, C. A., Gibbs J., Rein, G. L. (1991) "Groupware: some issues and experiences". Communications of the ACM, v.34, n.1, p.38-58.

Erl, T. (2005) "Service-Oriented Architecture – Concepts, Technology, and Design", Prentice Hall PTR.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. V., Loingtier, J., Irwin, J. (1997) "Aspect-Oriented Programming", In: Proc. of European Conference on Objective-Oriented Programming (ECOOP), Finlândia, Springer-Verlag LNCS 124.

Kiczales, G. et al. (2001) "Getting Started with AspectJ". In: Communications of the ACM, v. 44, n.10, p.59-65.

Maciel, R. S. P., Ferraz, C. G., Rosa, N. S. (2005) "InterDoc: Reference Architecture for Interoperable Services in Collaborative Writing Environments", In: Proc. of 9th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2005), p. 289-295, Coventry.

Maciel, R. S. P., David, J. M. N. (2007) "WGWSOA: A Service-Oriented Middleware Architecture to Support Groupware Interoperability". In: Proc. of the 11th International Conference on CSCW in Design (CSCWD), p. 556-561, Austrália, Apr.

Pinheiro, M. K., de Lima, V. J., Borges, M. R. S. (2003) "A framework for awareness support in groupware systems", Computers in Industry, Vol. 52 , Issue 1, Sept., p. 47 – 57.

Prinz, W. (1999) "NESSIE: An Awareness Environment for Cooperative Settings". In: Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work (ECSCW'99), pp. 391-410, Copenhagen, Denmark. Kluwer Academic Publishers, Dortrecht, NL, Sept. 12-16.

Schantz, R., Schmidt, D. (2001) "Middleware for Distributed Systems: Evolving the Common structure for Network-centric Applications", Encyclopedia of Software Engineering, Wiley & Sons.

Toledo, G. D. G., Bastos, D., David, J. M. N., Maciel, R. S. P. (2008) "Apoiando a Interoperabilidade entre as Atividades de Coordenação em uma Infra-estrutura de *Groupware*". V Simpósio Brasileiro de Sistemas Colaborativos (SBSC), Vila Velha – ES, p. 34-44.