3
Trabalhos Relacionados

Com o objetivo de entender e levantar os diversos problemas de arquiteturas distribuídas, foram analisadas diversas iniciativas e sistemas para a criação de aplicações baseadas em serviços. Os trabalhos relacionados foram divididos em três categorias: (i) trabalhos com arquitetura semelhante (peer-to-peer) (ii) trabalhos envolvendo semântica em serviços (iii) trabalhos envolvendo serviços pessoais e (iv) trabalhos sobre composição de serviços. Esta divisão teve como objetivo facilitar o entendimento sobre as diversas abordagens existentes para cada um dos aspectos de implementação das aplicações baseadas em serviços.

É importante ressaltar a questão da semântica nos serviços, que ainda permeia uma série de discussões sobre implementação. Existem diversas propostas para agregar semântica aos serviços disponibilizados pelas aplicações e poucas aplicações reais que façam uso desses mecanismos para aumentar a interoperabilidade.

3.1.
Sistemas Peer-To-Peer

3.1.1.
JXTA

JXTA é um projeto de pesquisa liderado pela Sun Microsystems que objetiva desenvolver uma arquitetura peer-to-peer independente de linguagem e de plataforma. Ele pretende criar uma plataforma comum que torne simples a construção de uma larga variedade de aplicações distribuídas nas quais múltiplos dispositivos são endereçados como partes (peers), e onde os mesmo podem se conectar em diferentes domínios (JXTA).

O núcleo não provê nenhuma aplicação, representando apenas a infra-estrutura para a construção de redes p2p. Ele consiste de um conjunto de componentes que permitem a criação, gestão e monitoramento de grupos de parceiros (peers) e de canais de comunicação. A partir deste conjunto básico de componentes, desenvolvedores de aplicação pode criar ambientes peer-to-peer sem ter que se

preocupar com segurança, protocolos de rede, mecanismos de troca de mensagens e outras funcionalidade básicas de redes P2P. O projeto é baseado em um conjunto de protocolos totalmente assíncronos (JXTA, 2001). Ele compreende uma infra-estrutura p2p completa que é capaz de buscar e compartilhar recursos entre parceiros (peers) JXTA.

O compartilhamento de informações nessas redes JXTA é possível por meio de diversos protocolos de troca de mensagens entre os parceiros. Apesar de cada protocolo JXTA ser público, não são considerados padrões da indústria como os padrões de Web Services e CORBA, por exemplo. Isto limita a interoperabilidade com software fornecido por outras empresas, o que se constitui em uma séria desvantagem.

Por fim, é importante ressaltar o aspecto central da camada de segurança do JXTA. Ela isola a segurança de detalhes de implementação, provendo comunicações seguras sem introduzir esforço adicional por parte dos desenvolvedores da aplicação.

3.1.2. Groove Network

Groove.Net é uma aplicação peer-to-peer para trabalho colaborativo. A plataforma Groove funciona em cima de uma série de protocolos proprietários que permitem a troca de informações entre os diferentes peers na rede. Os dados são sincronizados constantemente, fazendo com que cada peer tenha uma cópia atualizada dos dados compartilhados localmente. Isto garante que nenhum pedaço de informação fique indisponível no caso de algum peer não estar presente na rede.

O objetivo do sistema é permitir o compartilhamento de informações entre diversos usuários conectados em uma rede peer-to-peer. A política de sincronização dos dados constante implica em um overhead de performance, uma vez que todos os peers precisam replicar as informações em toda a rede.

Do ponto de vista de integração de dados, o Groove disponibiliza uma série de APIs (Application Programming Interfaces) que permitem que os serviços sejam integrados e acessados por outras aplicações.

3.1.2.1. Groove Web services

Recentemente, a Groove lançou o Groove Web Services, um conjunto de serviços que podem ser acessados utilizando os protocolos padrão de Web Services. Estes serviços permitem a integração do Groove com outros sistemas através de padrões da indústria. Através dos serviços Groove, é possível acessar as principais funcionalidades do sistema para compartilhar informações.

3.2. Semântica em serviços

3.2.1. DAML-S

DAML-S é uma iniciativa da DARPA para o desenvolvimento de uma especificação que contemple semântica nos serviços. Tendo em vista que a criação de linguagens de marcação para Ontologias, como (DAML+OIL), (OWL) e (SHOE), são um ponto fundamental para o desenvolvimento da Web Semântica, DAML-S é uma tentativa de criar uma ontologia baseada em DAML+OIL voltada para Web Services.

Segundo seus autores, WSDL é uma linguagem que contempla o conteúdo das mensagens que são trocadas assim como os protocolos para envio e recebimento enquanto que DAML-S procura expressar que tipo de informação está sendo trocado e o porquê desta troca. A partir destas informações, Web Services tornam-se interpretáveis por máquinas, permitindo a realização de diversas operações por agentes de software e o surgimento de propriedades inerentes à Web Semântica, como:

- *Descoberta*: localização de serviços que ofereçam uma funcionalidade específica e que atendam a um conjunto de restrições
- *Chamada*: acionamento de um serviço por um agente de software ou outro serviço.
- *Interoperabilidade*: aumentando a interoperabilidade das aplicações por meio do uso de semântica
- *Composição*: desenvolvimento de novos serviços a partir de outros existentes utilizando seleção automática e composição dinâmica.

- *Verificação*: de propriedades do serviço
- *Monitoramento da Execução*: acompanhamento de tarefas complexas realizadas por um conjunto de serviços, identificando falhas e problemas no caminho de execução.

Visando permitir estas operações, DAML-S define uma *Upper Ontology* para Web Services, que provê os mecanismos básicos para descrição de serviços. A partir desta ontologia, é possível criar uma taxonomia de serviços, permitindo a especificação de serviços para diferentes domínios através da criação de novos nós na hierarquia de classes pertencente à taxonomia. Um exemplo de descrição de serviços utilizando DAML-S pode ser visto a seguir.

```
<daml:Ontology>
 <daml:versionInfo>
...
 <service:presentedBy rdf:resource="http://www.daml.org/services/daml-
s/0.7/BravoAirService.daml#BravoAir_ReservationAgent" />
        …
 <profile:has_process rdf:resource="http://www.daml.org/services/daml-
s/0.7/BravoAirProcess.daml#BravoAir_Process" />
 <profile:serviceName>BravoAir_ReservationAgent</profile:serviceName>
 <profile:textDescription>This service…</profile:textDescription>
<profile:contactInformation>
<profile:Actor rdf:ID="BravoAir-reservation">
        <profile:name>BravoAir Reservation department</profile:name>
        …
 </profile:Actor>
 </profile:contactInformation>
 <profile:serviceParameter>
<profile:GeographicRadius rdf:ID="BravoAir-geographicRadius">
 <profile:serviceParameterName>BravoAir Geographic
Radius</profile:serviceParameterName>
 <profile:sParameter rdf:resource="http://www.daml.org/services/daml-
s/0.7/Country.daml#UnitedStates" />
 </profile:GeographicRadius>
 </profile:serviceParameter>
<profile:qualityRating>
<profile:QualityRating rdf:ID="BravoAir-goodRating">
 <profile:ratingName>SomeRating</profile:ratingName>
 <profile:rating rdf:resource="http://www.daml.ri.cmu.edu/ont/DAML-
S/concepts.daml#GoodRating" />
 </profile:QualityRating>
```

```
  </profile:qualityRating>
<profile:serviceCategory>
<profile:NAICS rdf:ID="NAICS-category">
        <profile:value>Airline reservation services</profile:value>
        <profile:code>561599</profile:code>
</profile:NAICS>
</profile:serviceCategory>
<profile:input>
<profile:ParameterDescription rdf:ID="DepartureAirport">
  <profile:parameterName>DepartureAirport</profile:parameterName>
  <profile:restrictedTo rdf:resource="http://www.daml.ri.cmu.edu/ont/DAML-
S/concepts.daml#Airport" />
  <profile:refersTo rdf:resource="http://www.daml.org/services/daml-
s/0.7/BravoAirProcess.daml#departureAirport_In" />
  </profile:ParameterDescription>
  </profile:input>
<profile:input>... </profile:input>
        ...
 <profile:output>
<profile:ParameterDescription rdf:ID="ReservationNumber">
        <profile:parameterName>FlightItinerary</profile:parameterName>
        <profile:restrictedTo rdf:resource="http://www.daml.ri.cmu.edu/ont/DAML-
S/concepts.daml#FlightItineraryList" />
<profile:refersTo rdf:resource="http://www.daml.org/services/daml-
s/0.7/BravoAirProcess.daml#FlightItineraryList" />
  </profile:ParameterDescription>
  </profile:output>
<profile:effect>
<profile:ParameterDescription rdf:ID="HaveFlight">
        <profile:parameterName>HaveFlight</profile:parameterName>
<profile:restrictedTo rdf:resource="http://www.daml.ri.cmu.edu/ont/DAML-
S/concepts.daml#HaveFlightSeat" />
<profile:refersTo rdf:resource="http://www.daml.org/services/daml-
s/0.7/BravoAirProcess.daml#HaveFlightSeat" />
  </profile:ParameterDescription>
  </profile:effect>
  </profileHierarchy:AirlineTicketing>
```

Figura 5 – Exemplo de uma descrição de serviço em DAML-S

Uma das barreiras para a adoção de DAML-S é a necessidade de uma nova estrutura para descrição e chamada aos serviços. Muito do trabalho realizado em

torno de WSDL e UDDI teria de ser refeito para adequar-se à especificação de DAML-S. Visto que já existem diversos projetos e serviços que adotam WSDL e UDDI como padrão para descrição dos serviços e como infra-estrutura para busca de serviços, a necessidade de migrar toda essa infra-estrutura para outra correspondente pode dificultar a adoção de DAML-S em diversos casos.

3.2.2. Web Services Modeling Framework (WSMF)

WSMF (Fensel & Bussler, 2002) é uma proposta ainda em fase inicial desenvolvida por um grupo de empresas e universidades na Europa. WSMF, assim como a proposta apresentada neste trabalho, está extremamente baseada no uso de serviços para a criação de aplicações na Web Semântica.

Por outro lado, a composição de ontologias é limitada ao uso de mediadores e a composição de serviços é baseada em uma nova estrutura também proposta no projeto. Nenhum dos mecanismos de composição e de desenvolvimento de serviços já existente é aproveitado.

O framework WSMF enumera oito camadas para uma composição de serviços bem sucedida. São elas:

- Tipos: Descreve os dados trocados
- Semântica: Apresenta como os dados trocados devem ser populados para que sejam válidos no contexto do serviço.
- Transporte: Representa o protocolo de transporte que deve ser utilizado por ambas as partes para o correto fornecimento do serviço.
- Seqüência de troca: Define as políticas de troca de mensagens que impedem que uma mensagem seja recebida ou enviada mais de uma vez
- Definição de processo: Detalha o fluxo de mensagens trocadas para a execução de um processo
- Segurança: Provê a capacidade de criptografar mensagens para garantir que seu conteúdo não seja alterado por software malicioso e que haja não-repudiação para ambas as partes
- Sintaxe: As informações precisam ser apresentadas em uma sintaxe comum. Hoje em dia, XML é a alternativa mais comum.

- Configurações específicas de parceiros: Realiza ajustes específicos em função de necessidades de parceiros

A partir destas camadas, o framework pode instanciar a coordenação de diferentes serviços a partir das especificações. O desenvolvimento de WSMF ainda está sendo feito e possivelmente será feita uma extensão de DAML-S para contemplar aspectos de coordenação não definidos em DAML-S.

3.2.3. TAP

Tap (TAP) é um projeto desenvolvido na Universidade de Stanford que visa alavancar o desenvolvimento da Web Semântica através de três pontos básicos: Padrões e Métodos, Aplicações e Protocolos.

O ponto central do Projeto TAP é a introdução do conceito de Reference By Description. Através de uma série de padronizações sobre a forma de descrição de descrição de um conceito, é possível que diversas ontologias sejam compostas em uma só através de pontos de contato. Estes são passados a cada uma das entidades detentoras das ontologias como uma descrição baseada em um padrão definido na TAP KB. Esta é uma base de conhecimento que contém formas genéricas de representar conhecimento sobre os conceitos mais presentes em aplicações pela Web como artistas, empresas, pessoas, músicas, eventes e esportes, entre outros.

A partir destas descrições-padrão, as diversas aplicações podem então compor suas informações para formar uma grande base de conhecimento.

3.2.4. Retsina Calendar Agent

O Retsina Semantic Web Calendar Agent permite a interoperabilidade entre calendários baseados em descrições RDF e sistemas de gestão de informações pessoais (PIM) como o Microsoft Outlook. Calendários e eventos podem ser descritos na Web em RDF, utilizando ontologias como a Hybrid ical RDF-Schema ou a DAML Agenda e ligadas às informações de contato dos usuários descritas em suas homepages. O agente Retsina consiste em um sistema de agendamento distribuído de compromissos e do parser de calendários descritos através de ontologias. O agente auxilia os usuários na organização e na marcação de

compromissos entre diferentes indivíduos, a partir dos calendários armazenados pelo MS Outlook.

3.3. Serviços pessoais

3.3.1. Net MyServices (HailStorm)

Como parte da iniciativa .Net™, a Microsoft™ anunciou o desenvolvimento de My Services, um conjunto de serviços básicos disponíveis 24 horas por dia. A plataforma MyServices hospedaria um conjunto de serviços de informações pessoais, permitindo que desenvolvedores de aplicações acessem dados para personalizar suas aplicações.

O principal problema desta plataforma é sua arquitetura no estilo Application Service Provider (ASP). Os usuários da plataforma teriam que deixar suas informações pessoais sob a custódia da Microsoft™, algo que levanta diversas questões de privacidade além de outras limitações no acesso. Qualquer desconexão da rede por parte dos servidores MyServices deixaria os usuário isolados de seus serviços e dados pessoais.

Alguns dos serviços incluídos inicialmente pela Microsoft na plataforma MyServices incluem: perfil, calendário, caixa de entrada, configurações de aplicação, documentos e presença. Uma aplicação baseada em serviços simples permitiria que usuários buscassem documentos pessoas a partir de um telefone celular usando seu serviço de documentos e mandassem para amigos ou colegas de trabalho.

Com relação às questões de autenticação, o acesso ao .Net MyServices é feito utilizando o Microsoft Passport, que implementa o algoritmo KERBEROS . Os usuários precisam possuir credenciais para acessar um serviço específico.

Como forma de descrever as informações manipuladas pelos serviços no .Net MyServices, a linguagem HSDL (MyServices Data Manipulation Language) foi criada. HSDL é focada no transporte dos dados para dentro e para fora dos serviços e de seus dados associados. Seu sintaxe reflete ainda outras questões relacionadas à implementação do .Net MyServices, como segurança e estrutura de dados (Mahamy et al., 2001).

A partir dessa base, cada serviço é estruturado a partir de nós XML chave que possuem mais relevância do que outros. As interações do MyServices envolvem a troca de mensagens SOAP contendo dados XML baseados em HSDL. Linguagens baseadas em XML permitem uma representação fácil e flexível dos dados fornecidos pelos serviços. No entanto, implicam em um processamento extra de XML nos clientes dos serviços.

3.3.2. Apple .Mac™

Apple .Mac é um serviço fornecido pela Apple que permite aos usuários do Macintosh compartilhar seus dados através da Internet. Algumas aplicações do Mac como o iCal e iDisk são capazes de publicar os dados no .Mac automaticamente.

Usuários podem enviar URL contendo informações pessoais para amigos ou colegas de trabalho através de e-mail. Essas URLs são geradas automaticamente pelo .Mac quando o usuário publica seus dados pessoais na Web.

Apesar da interface simples de utilização, o sistema possui uma limitação severa: sua interface para as informações pessoais. Os serviços .Mac são apresentados em páginas HTML e precisam ser acessados através de http. Ele não permite que qualquer outra aplicação acesse recursos usando qualquer tipo de middleware. A informação somente é apresentada para os usuários finais no web site do .Mac. Outras partes ou parceiros não tem acesso aos componentes da aplicação, o que os torna incapazes de atualizar informações ou coordenar processos entre máquinas diferentes.

3.4. Composição de Serviços

3.4.1. BPEL

BPEL (BPEL) é um passo importante na busca de desenvolver aplicações em um domínio com processos de negócios flexíveis, com unidades distribuídas, independentes de plataforma. Esta especificação é uma linguagem baseada em XML que explicita a lógica do fluxo de processos de negócios através de um conjunto de atividades de um processo e do controle de dependências entre estas atividades. A

princípio, um processo especificado em BPEL explicita suas funcionalidades como um serviço e utiliza outros serviços para disponibilizar determinada funcionalidade.

Abaixo serão destacados os principais blocos semânticos de uma especificação em BPEL.

```
<process name="purchaseOrderProcess"
         targetNamespace="http://acme.com/ws-bp/purchase"
         xmlns="http://schemas.xmlsoap.org/ws/2002/07/business-process/"
         xmlns:lns="http://manufacturing.org/wsdl/purchase">          ...
</process>
```

Figura 3 - Cabeçalho de definição de processo de coordenação

```
   <partners>
      <partner name="customer"
               serviceLinkType="lns:purchaseLT"
               myRole="purchaseService"/>
      <partner name="invoiceProvider"
               serviceLinkType="lns:invoiceLT"
               myRole="invoiceRequester"
               partnerRole="invoiceService"/>
      <partner name="shippingProvider"
               serviceLinkType="lns:shippingLT"
               myRole="shippingRequester"
               partnerRole="shippingService"/>
      <partner name="schedulingProvider"
               serviceLinkType="lns:schedulingLT"
               partnerRole="schedulingService"/>
   </partners>
```

Figura 4 - Definição de participantes do processo de coordenação

```
   <containers>
      <container name="PO" messageType="lns:POMessage"/>
      <container name="Invoice"
                 messageType="lns:InvMessage"/>
      <container name="POFault"
                 messageType="lns:orderFaultType"/>
      <container name="shippingRequest"
                 messageType="lns:shippingRequestMessage"/>
      <container name="shippingInfo"
                 messageType="lns:shippingInfoMessage"/>
      <container name="shippingSchedule"
                 messageType="lns:scheduleMessage"/>
   </containers>
```

Figura 5 - Containers são canais para a troca de informações entre atividades

```
<faultHandlers>
   <catch faultName="lns:cannotCompleteOrder"
          faultContainer="POFault">
      <reply   partner="customer"
               portType="lns:purchaseOrderPT"
               operation="sendPurchaseOrder"
               container="POFault"
               faultName="cannotCompleteOrder"/>
   </catch>
</faultHandlers>
```

Figura 6 - Tratadores de exceção utilizados na especificação do processo de coordenação de componentes

```
<sequence>
   <receive partner="customer"
            portType="lns:purchaseOrderPT"
            operation="sendPurchaseOrder"
            container="PO">
   </receive>
   <flow>
      <links>
         <link name="ship-to-invoice"/>
         <link name="ship-to-scheduling"/>
      </links>
      <sequence> ... </sequence>
      <sequence> ... </sequence>
      <sequence> ... </sequence>
   </flow>
   <reply partner="customer"
          portType="lns:purchaseOrderPT"
          operation="sendPurchaseOrder"
          container="Invoice"/>
</sequence>
```

Figura 7 - Fluxo de atividades para a coordenação de componentes

De forma complementar à linguagem BPEL, surgiram outras duas propostas para o suporte a coordenação de serviços e o suporte ao controle da ocorrência de transações entre serviços, o WS-Coordination e o WS-Transaction respectivamente.

A primeira (WSCOORD) descreve um framework extensível para prover protocolos que coordenam atividades de aplicações distribuídas. Ao utilizar WS-C, aplicações distribuídas, operando em ambiente heterogêneos, podem (1) criar um contexto necessário para a propagação de uma atividade para outros serviços e (2) se registrar para a participação em protocolos de coordenação.

Já a segunda especificação (WSTRANS) descreve dois tipos de coordenação, a Transação Atômica (TA) e Atividade de Negócio (AN), que são utilizadas em conjunto com WS-Coordination. O tipo TA é utilizado para coordenar atividades que tenham curta duração. O tipo de coordenação AN é utilizado para atividades de coordenação de longa duração ou caso seja necessário aplicar alguma lógica de negócio para o tratamento de exceções que ocorram durante a execução de algum processo.

BPEL, WS-Coordination, e WS-Transaction são utilizadas em conjunto com WSDL e UDDI e podem prover a base para um novo modelo de programação, onde o dinamismo da evolução dos processos de negócio pode requerer uma maior suporte a estas características no processo de geração ou adaptação de aplicações.

3.4.2. BTP

Em março de 2001, a Oasis (Organization for the Advancement of Structured Information Standards) criou um comitê para elaboração de um protocolo baseado em XML para realização de transações envolvendo e-business na Internet. Este grupo é formado por diversos membros de indústria de informática.

Apesar do foco em protocolos de transação, o comitê adotou Web Services como componentes distribuídos envolvidos nas transações. Diversos membros enviaram propostas diferentes para criação do protocolo. Entretanto, apenas uma delas será discutida nesta seção, Business Transactions Protocol (Little, 2001).

A idéia fundamental do Business Transactions Protocol (BTP) é criar um modelo de transações que seja extensível para dar suporte aos diferentes tipos de transações que podem ocorrer na Web.

Esta premissa tem como motivação os diversos tipos de transações possíveis que podem ocorrer quando da composição de Web Services, o que torna o modelo flat ineficiente.

Dessa forma, o BTP propõe a criação de um framework que possa dar suporte à criação de diversos modelos estendidos de transações, permitindo o reuso de mecanismos comuns de controle das transações.

Através deste framework, aplicações integradoras poderiam utilizar modelos transacionais de acordo com as características das operações envolvidas. De acordo com a necessidade, diversas subtransações poderiam ser agrupadas em transações maiores, em um modelo semelhante ao de transações aninhadas. Em caso de transações de curta duração, o modelo de transações planas poderia ser empregado.

No entanto, diversos modelos diferentes causariam incompatibilidade entre serviços do ponto de vista transacional. Por este motivo, os mecanismos de descoberta de serviços teriam de ser estendidos de forma a permitir uma negociação para estabelecimento de um protocolo de transações aceito por todas as partes.

Esta abordagem é interessante por permitir que cada aplicação utilize seu próprio protocolo de transações. Entretanto, os integradores de serviços ficam presos ao framework de criação, o que pode diminuir o nível de interoperabilidade dos serviços envolvidos. Idealmente, as aplicações integradoras deveriam funcionar sem necessidade de componentes adicionais aos componentes básicos da arquitetura de Web Services.