2
Trabalhos Relacionados

Nesta seção são analisados trabalhos encontrados na literatura relacionando-as com a proposta deste trabalho. Foram considerados trabalhos cuja abordagem visasse a construção de sistemas autoadaptativos que permeiam os conceitos de disponibilidade e interoperabilidade na troca de informações entre o sistema e o serviço web. Para solucionar o desafio da disponibilidade dos serviços web, os trabalhos analisados utilizaram de mecanismos de regras e componentes como meio de prover autonomia ao sistema. Já para superar o desafio da interoperabilidade os trabalhos relacionados explorados utilizaram o conceito de sistema multiagentes para assegurar a autonomia ao *software*.

2.1.
A Rule Based Approach For Availability Of Web Services

O trabalho de Liang *et al* (LIANG, LAM, *et al.*, 2008) apresenta um *framework* para tratar a questão de disponibilidade de serviços web através de um mecanismo para seleção de serviço substituto. O *framework* é composto de duas grandes etapas: "Term Categorization Tool", responsável por categorizar a lista de serviços substitutos e "Rule-Based Service Discover", responsável por selecionar o novo serviço.

A etapa "Term Categorization Tool" é composta de sete atividades: "*Document Parser", "Relationship Dimension Selector", "Pre-processor", "Term Stemming", "Co-occurrence Model", " Similarity Probabilistic Model" e "Categorization Result Merging",* descritas em mais detalhes a seguir.

A primeira atividade, o *Document Parser* é responsável por extrair os dados dos arquivos de descrição de serviços que serão analisados. As descrições dos serviços podem utilizar a notação WSDL ou OWL-S. O próximo passo é o *Relationship Dimension Selector* que tem como atribuição identificar o tipo de descritor de serviço que foi utilizado, suas principais classes e atributos. A terceira atividade a ser executada é o *Pre-processor*, encarregado de realizar um filtro

sobre o conjunto dos termos utilizados nos descritores de serviços capturando os termos comuns a todos os descritores. Os termos que não forem considerados relevantes serão descartados. Logo a seguir, o *Term Stemming* rotula os serviços web categorizando os termos de acordo com a convenção UNSPSC[1], que foi elabora pela *United Nations Standard Products and Services Code*.

Os próximos passos são os mecanismos de similaridade e o mecanismo de junção dos dois resultados obtidos dos cálculos da similaridade. A primeira classificação é realizada pelo *Co-occurrence Model* cujo dispositivo de graduação dos serviços se baseia numa matriz de frequências. Já o *Similarity Probabilistic Model* (SMV) é o segundo mecanismo utilizado para categorização dos serviços candidatos a substituto. Este mecanismo faz uso de um algoritmo de similaridade para classificar os serviços web. Por fim, o *Categorization Result Merging* possui a atribuição de unir os resultados obtidos da categorização dos serviços web pelos mecanismos de co-ocorrência e similaridade, uma vez que, ao se utilizar dois mecanismo para classificação dos serviços web, estes podem atribuir ordens diferentes para o mesmo serviço.

Ao final da atividade "Term Categorization Tool" tem-se uma lista de candidatos a serviço substituto categorizado pelo seu nível de similaridade com o serviço que falhou. Esta lista será repassada à atividade "Rule-Based Service Discover", descrita a seguir.

A etapa "Rule-Based Service Discover" é composta de quatro atividades: "*Service Attribute Selection", "Rule Identification", "Rule Evaluation" e "Service Subtitute Identification",* descritas em mais detalhes a seguir.

O *Service Attribute Selection* é responsável por selecionar os serviços web que serão avaliados. Já o *Rule Identification* identifica a regra de equivalência que será utilizada para avaliar o serviço web. O conjunto serviço e regra de avaliação são então encaminhados ao *Rule Evaluation* que é responsável por avaliar a equivalência do serviço selecionado em relação à lista de termos categorizados. No sistema de regras proposto, existem três possibilidades de equivalência: a equivalência mais forte ocorre quando o serviço é exatamente o mesmo, porém fornecido por outro provedor, o segundo caso é o novo serviço ser um subserviço

[1] A UNSPSC fornece um padrão aberto, global e multissetor para classificação eficiente e precisa de produtos e serviços (UNSPSC, 2012).

do original e terceiro e último caso de equivalência ocorre encontrando um serviço com características similares. Por fim, o *Service Subtitute Identification* é o incumbido de informar ao sistema o novo serviço web.

O *framework* elaborado por Liang *et al* utiliza um conjunto de regras e dois mecanismos de similaridade para avaliar e selecionar um novo serviço que substitua o serviço que falhou. Nosso trabalho utiliza o conceito de computação autonômica aplicada através do conceito de sistemas multiagentes para superar o desafio da disponibilidade da troca de informações entre o dispositivo móvel e o serviço web. Adicionalmente, nosso *framework* também possui mecanismo para superar as dificuldades de interoperabilidade que podem surgir no momento da troca de serviços, que não é tratado no trabalho de Liang *et al*.

2.2. Adding High Availability And Autonomic Behavior To Web Services

Este trabalho (BIRMAN, RENESSE e VOGELS, 2004) apresenta uma extensão para a atual arquitetura dos Serviços Web proposta pela W3C para que estes deem suporte ao conceito de disponibilidade, ou seja, que o sistema supere questões relativas a falhas de comunicação. A extensão construída por Birman *et al*, que é composta de cinco componentes: "Component Health Monitoring", "Consistent and Reliable Messaging", "Data Dissemination", "Monitoring and Distributed Control" e "Event notification", descritos a seguir.

O *Component Health Monitoring* (CHM) é responsável por monitorar determinados componentes do sistema e qualquer modificação de estado destes componentes deve ser comunicado a todos os outros componentes "interessados" nesta mudança. Já o *Consistent and Reliable Messaging* (CRM) estabelece um mecanismo de comunicação entre componentes através interfaces de grupo e protocolo TCP. Desta forma, a replicação da informação entre os componentes ocorre de forma dinâmica e segura, permitindo a sincronização de todos os componentes com o mesmo endereço TCP do serviço ativo.

Estes dois componentes podem ser implementados tanto pelo cliente quanto pelos provedores de serviço. Quando utilizado pelos provedores, se obtém serviços capazes de se redirecionar para outros serviços ativos ou mesmo capazes de reiniciar os componentes que apresentam problema.

Visando o novo paradigma da computação autonômica, três componentes foram construídos propondo certo nível de autonomia à extensão proposta. A seguir temos a descrição destes componentes:

O *Data Dissemination* (DD) provê mecanismo de disseminação de informação a todos os clientes que possuem ligações diretas através de múltiplas conexões. O DDS padroniza a troca de informação uma vez que as mensagens são enviadas a grupos de clientes, no entanto permite extensões através de plug-in para comunicação com protocolos específicos.

O *Monitoring and Distributed Control* (MDC) fornece um mecanismo para monitorar uma coleção de recursos distribuídos, reportando os serviços de melhor qualidade de informação. Dentre suas atribuições está também a confecção de uma hierarquia de domínios, atualizada em tempo real, contendo os caminhos para cada serviço.

O *Scalable Event Notification* (EVN) é um módulo ainda em evolução, responsável por monitorar uma grande quantidade de informações presentes nos servidores. Quando uma condição previamente acordada com um componente ocorrer ou estiver próxima de ocorrer, este módulo deverá pró-ativamente notificar o componente da possível materialização daquele evento.

Por incorporar certo nível de autonomia aos seus componentes através do módulo ENV, onde há monitoramento de eventos para prevenção de falhas, nota-se que este trabalho possui alinhamento com o conceito de computação autonômica ao tratar a questão da indisponibilidade de serviços web. No entanto, tange um ponto controverso sobre a modificação da arquitetura elaborada pela W3C (W3C, 2004) para construção de serviço web, que hoje é bem aceita. Nosso trabalho aborda a questão de disponibilidade de serviços através de um mecanismo reativo para tratar falhas de comunicação, também visto como computação autonômica, sem a necessidade de modificar a arquitetura elaborada pela W3C. O conceito de autonomia aplicado em nosso trabalho é implementado através de um sistema multiagentes, em que os agentes possuem habilidades de cooperação, colaboração e negociação, tornando-os capazes de se comunicar com outros agentes a fim de atingir seu objetivo de superar a indisponibilidade do serviço web.

2.3. A Framework For Integrating Web Services And Multi-Agent Systems

Hemayati *et al* (HEMAYATI, MOHSENZADEH, *et al.*, 2010) elabora uma arquitetura baseada no conceito de sistemas multiagentes em que o agente chamado *Broker* possui a habilidade de traduzir as mensagens trocadas entre entidades de padrões de comunicação distintos, neste caso, agentes e serviços web. O agente *Broker* é capaz de traduzir do padrão ACL para SOAP e vice-versa e também do padrão DF para UDDI e vice-versa. Os padrões ACL e DF são utilizados pelos agentes enquanto que SOAP e UDDI são utilizados pelos serviços web.

Este *framework* é capaz de lidar tanto com requisições de informações quanto publicações de serviços, esta duas atividades seguem caminhos similares no *framework* para tradução das informações entre os padrões agentes e serviços web. A seguir apresentamos o detalhamento das seis atividades – "*Identifier*", "*Syntax Analyzer*", "*Converters*", "*SOAP 2 Data Structure*", "*Data Structure Creator*", "*Treasure"* e "*Non-functional/Functional Module*" - executadas pelo *framework* para assegurar a interoperabilidade na troca de informações entre o agente e o serviço web.

O *Identifier* é a primeira atividade e tem como atribuição identificar o formato e tipo de requisição. Os formatos podem ser ACL, SOAP ou linguagem natural. Os tipos são requisição ou publicação. As próximas etapas referem-se ao tipo requisição de informação.

O *Syntax Analyzer* é executado quando a requisição vier em formato de linguagem natural. Este analisador possui um mecanismo para processamento da linguagem natural onde são identificados atributos funcionais e não funcionais da requisição. Caso não seja requisição em formato de linguagem natural, ou seja, vier em formato ACL ou SOAP os *Converters* serão executados. Se a informação recebida vier em formato ACL, esta é transformada em SOAP através da funcionalidade "ACL2SOAP" e enviada para o extrator de informações "*SOAP 2 Data Structure*". Se a informação vier em formato SOAP, esta é enviada diretamente ao "*SOAP 2 Data Structure*". Já o *SOAP 2 Data Structure* é responsável por extrair as especificações das requisições e encaminhar ao "*Data Structure Creator*".

O *Data Structure Creator* recebe uma relação pai-filho entre os atributos não funcionais da requisição do módulo "*Treasure*" e elabora uma estrutura de dados para publicação dos atributos não funcionais dos agentes e/ou serviços web. Em seguida a estrutura é encaminhada para o módulo "*Non-functional/Functional Module".*

O *Treasure* é responsável por identificar os termos equivalentes contidos na informação a ser transmitida e os termos padrões adotado pelos *frameworks*. Este módulo também é responsável por identificar os atributos necessários a serem preenchidos em cada protocolo de comunicação. Após esse mapeamento, as informações são repassadas ao módulo "*Non-functional/Functional Module*".

O *Non-functional/Functional Module* identifica o protocolo a ser utilizado, no caso ACL ou SOAP, e envia as informações ao tradutor "*Converters*" para que a mensagem seja codificada no protocolo necessário. Após a conversão correta da mensagem esta é encaminha ao seu destinatário.

Este *framework* permite que agentes publiquem seus serviços nos diretórios DF dos agentes e UDDI dos serviços web. Para fornecer este serviço, os autores observaram a necessidade em manter os registros DF e UDDI sempre sincronizados, o que ocasiona dois registros replicados no DF e UDDI. Também é possível que agentes e serviços web procurem serviços nos diretórios DF e UDDI. Para obter melhor integração dos componentes de nosso trabalho utilizamos o *framework* WSIG, baseado em agentes JADE e também elaborado pela Telecom Italia SpA (JADE, 2011), para transpor o desafio da interoperabilidade. Além deste desafio, nosso trabalho também abrange a questão da indisponibilidade do serviço web que não é tratado neste trabalho de Hemayati *et al*.

2.4. Bridging Multi Agent Systems and Web Services: Towards Interoperability Between Software Agents and Semantic Web Services

A principal inovação proposta por Shafiq *et al* (SHAFIQ, DING e FENSEL, 2006) é a utilização de um agente, chamado de *AgentWebGateway*, capaz de traduzir as informações trocadas entre agentes de software e serviços web. Este agente trabalha como um sistema intermediário que realiza a conversão das mensagens do padrão agente para o padrão serviço web e vice-versa. Desta forma,

consegue-se a interoperabilidade técnica na troca de informações entre serviços com protocolos de dados distintos.

O sistema proposto neste trabalho é composto de três principais atividades. A primeira é a atividade de descobrir serviços, chamada de *Service Discovery Converter*, que permite a um agente procurar por serviços no diretório UDDI dos serviços web e vice-versa. Para realizar uma busca no registro UDDI, a mensagem enviada pelo agente será traduzida para uma mensagem no padrão dos serviços web e posteriormente o serviço será procurado no registro UDDI. O retorno da mensagem, no padrão utilizado pelos serviços web, deverá ser traduzido novamente para o padrão do agente e encaminhado ao mesmo.

Outra atividade executada pelo *AgentWebGateway* é publicação de serviços dos agentes nos diretórios dos serviços web e vice-versa. Para realizar esta tarefa, o agente utiliza o tradutor de descrição de serviço, chamada de *Service Description Converter*, que fará a conversão do padrão DF(agente) para WSDL(serviço web). Desta forma, um agente poderá publicar seus serviços tanto no diretório DF dos agentes quanto no diretório UDDI dos serviços web. Também é possível um serviço web publicar seus serviços no diretório DF dos agentes. Finalmente, para concluir a interoperabilidade na comunicação entre agentes e serviço web, o *AgentWebGateway* realiza a conversão dos protocolos de comunicação ACL(agente) e SOAP(serviço web). Através da atividade de "*Communication Protocol Converter*", o *AgentWebGateway* é capaz de converter mensagens ACL em mensagens SOAP e vice-versa.

Através destas três atividades tem-se um sistema capaz de realizar a comunicação completa entre os sistemas multiagentes e os serviços web, sem que seja necessária qualquer alteração nos atuais padrões de implementação dos agentes e serviços web. Em nosso trabalho utilizamos uma abordagem similar, proposta pela FIPA e implementada pelo JADE-LEAP através do *JadeGatewayAgent*. Este agente possui uma característica interessante que é a tradução automática de qualquer mensagem recebida do serviço web para protocolo utilizado pelos agentes. Nosso *framework* também contempla o suporte à questão de disponibilidade dos serviços, tema não tratado na proposta de Shafiq *et al*.

2.5. Discussão

Os trabalhos relacionados que foram encontrados na literatura abordam soluções para os desafios da indisponibilidade do serviço web (Liang, Birman) ou da interoperabilidade (Hemayati, Shafiq), mas em nenhum momento abordam os dois desafios simultaneamente. O comportamento autonômico utilizado para recuperação de falhas de disponibilidade do serviço web é apresentado por Liang *et al* através de um conjunto de regras consolidadas em um *framework*. Birman *et al* aborda o mesmo desafio elaborando uma arquitetura de componentes como extensão para a atual arquitetura dos serviços web proposta pela W3C. Já para o desafio da interoperabilidade, são construídas soluções baseadas no conceito de multiagentes. Hemayati *et al* e Shafiq *et al* elaboram agentes que se comportam como sistemas intermediários traduzindo as mensagens trocadas entre o sistema e o serviço web.

Vale lembrar que o objetivo principal deste trabalho é auxiliar no desenvolvimento de aplicações com capacidade de recuperação automática frente à indisponibilidade de serviços web, permitindo também interoperabilidade ao utilizar serviços alternativos. Nenhum dos trabalhos relacionados encontrados resolve estes dois desafios de uma só vez. Para isso, é proposto um *framework* baseado no conceito de computação autonômica, projetado como um sistema multiagentes para funcionar em ambiente de computação móvel utilizando-se de serviços web, que é apresentado em detalhes no capítulo seguinte.