Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

Leila Negris Bezerra

Uso de ontologia em serviço de contexto e descoberta de recursos para autoadaptação de sistemas

Rio de Janeiro
2011

Leila Negris Bezerra

Uso de ontologia em serviço de contexto e descoberta de recursos para autoadaptação de sistemas

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Redes de Telecomunicações e Sistemas Distribuídos.

Orientador: Prof. Dr. Alexandre Sztajnberg

Rio de Janeiro

2011

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

B574 Bezerra, Leila Negris.
Uso de ontologia em serviço de contexto e descoberta de recursos para autoadaptação de sistemas. / Leila Negris Bezerra - 2011.
162 f. : il.

Orientador: Alexandre Sztajnberg.
Dissertação (mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia Eletrônica. 2. Redes de computação - Dissertações. I. Sztajnberg, Alexandre. II. Universidade do Estado do Rio de Janeiro. Faculdade de Engenharia. III. Título.

CDU 621.38:004.7

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial desta tese, desde que citada a fonte.

_____ _____

Assinatura Data

Leila Negris Bezerra

Uso de ontologia em serviço de contexto e descoberta de recursos para autoadaptação de sistemas

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Redes de Telecomunicações e Sistemas Distribuídos.

Aprovada em: 13 de julho de 2011.

Banca examinadora:

_____
Prof. Dr. Alexandre Sztajnberg (Orientador)
Faculdade de Engenharia - UERJ

_____
Prof. Dr. Neide Santos
Faculdade de Engenharia - UERJ

_____
Prof. Dr. Orlando Gomes Loques Filho
Universidade Federal Fluminense - UFF

Rio de Janeiro
2011

DEDICATÓRIA

Aos meus pais e meus professores de todos os tempos.

AGRADECIMENTOS

Aos meus pais, minha família e amigos queridos, muito obrigada pela torcida e por entenderem a distância criada.

Quero agradecer de forma muito especial ao Rodrigo Costa de Alencar, Pedro Leonardo A. dos Santos, Rosane Guimarães e Aretuza Cesar Santos pelas tantas vezes e diferentes formas como me ajudaram durante o tempo de desenvolvimento deste trabalho.

Quero agradecer também ao meu orientador Alexandre Sztajnberg por todo o suporte que me deu, ao meu companheiro de mestrado Andre Felipe de Almeida Monteiro, pela parceria constante e aos meus gestores nas empresas onde tive a oportunidade de trabalhar durante o período do mestrado: Julio Marchiori e Leonardo Hiss além do Almir Barbosa e Julio Cesar Barbosa, pela liberação nas ocasiões em que precisei.

RESUMO

BEZERRA, Leila Negris. *Uso de ontologia em serviço de contexto e descoberta de recursos para autoadaptação de sistemas*. 160 f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia Eletrônica, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2011.

Aplicações cientes de contexto precisam de mecanismos para recuperar informações sobre o seu contexto de execução. Com base no contexto atual, tais aplicações são capazes de se autoadaptar para fornecer informações e serviços adequados aos seus usuários. A abordagem comum para infraestruturas de apoio às aplicações sensíveis ao contexto fornece serviços para a descoberta de recursos através da utilização de pares <chave-valor> e motores que executam apenas correspondência sintática. Esta abordagem não considera as possíveis relações semânticas entre as palavras-chave usadas. Portanto, a sua expressividade semântica limitada, leva a um serviço de descoberta que provê baixa taxa de recuperação e baixa acurácia. Este trabalho apresenta a utilização de uma outra abordagem para o serviço de contexto e descoberta, que utiliza ontologias para representar os recursos do contexto de execução e capturar a semântica da consulta do usuário, melhorando assim o processo de descoberta para a autoadaptação de sistemas sensíveis ao contexto. A abordagem proposta oferece também pontos de extensão para as aplicações clientes através da utilização de outras ontologias. Esta abordagem foi integrada à infraestrutura CDRF, de forma a adicionar semântica aos serviços desenvolvidos neste projeto. Exemplos de aplicações são também propostos para demonstrar a utilização dos novos serviços.

Palavras-chave: Ontologia; Aplicações cientes de contexto; Serviços de contexto e descoberta.

ABSTRACT

Context-aware applications demand ways of retrieving context information from the environment. Based on the current context, such applications are able to self-adapt to provide the correct information and services to its users. The usual approach for supporting infrastructures for context-aware applications provides facilities for resource discovery using <key-value> pairs and discovery engines that perform syntactic matching. This approach does not consider the possible semantic relations between the keywords used. So its limited semantic expressiveness often leads to poor discovery results. This paper presents the use of a different approach for service discovery that uses ontologies to represent resources and capture the semantics of the user's query, improving the discovery process for self-adaptation of context-aware systems. The proposed approach also offers extension hooks to the client applications through the use of other ontologies. This approach is integrated into the CDRF framework and adds semantics to the services developed in that project. Example applications are also proposed to demonstrate the use of the new services.

Keywords: Ontology; Context-aware applications; Context and discovery services.

LISTA DE CÓDIGOS

LISTA DE ILUSTRAÇÕES

SUMÁRIO

INTRODUÇÃO

A dependência de contexto nos serviços computacionais está relacionada à capacidade que determinada aplicação tem de utilizar informações de contexto para se autoadaptar.

Aplicações cientes ou sensíveis ao contexto (*context aware*) são aplicações capazes de alterar seu comportamento de acordo com análises realizadas sobre informações extraídas de seu ambiente de execução. A partir da obtenção de tais informações, através de fontes como entradas de usuários e monitoração de sensores, a aplicação pode determinar dinamicamente, os tipos de informações ou serviços que serão disponibilizados para seus usuários.

Considera-se contexto como “qualquer informação que possa ser utilizada para caracterizar o estado de uma entidade. Onde uma entidade representa uma pessoa, lugar ou objeto que seja considerado relevante para a interação entre o usuário e uma aplicação, incluindo o próprio usuário e a aplicação em questão” (DEY, 2000).

Existe uma grande variedade de tipos de contextos. Dentre eles, podemos citar: contextos físicos (localização, tempo), contextos ambientais (clima, níveis de luz e som), contextos informacionais (preços de ações, resultados desportivos), contextos pessoais (saúde, humor, cronograma de atividades), contextos sociais (atividade de grupo, relações sociais, quantidade de pessoas em uma sala), contextos de aplicação (*emails* recebidos, sites visitados) e contextos da operação dos sistemas (tráfego de rede, status de impressoras) (KORKEA-AHO, 2000).

Sistemas cientes de contexto fazem uso extensivo de diferentes tipos de recursos. Durante sua execução, estes sistemas estão sujeitos a mudanças constantes em sua operação. Para isto são necessários mecanismos para proporcionar a descoberta de serviços com base na situação atual (contexto) do usuário e da aplicação (BROENS, 2004).

Uma característica fundamental de sistemas cientes de contexto é a transparência no fornecimento de serviços. O processo de descoberta de serviços (recursos) relevantes e sua invocação devem ser encapsulados do ponto de vista dos usuários.

O processo de desenvolvimento de aplicações sensíveis ao contexto é de alta complexidade e usualmente demorado, devido à necessidade de manipulação de diversos contextos, provenientes de ambientes altamente dinâmicos, distribuídos, heterogêneos e em constantes mudanças.

Devido à grande quantidade de fontes de informação, torna-se complicada a organização das mesmas, e devido à gigantesca quantidade de dados produzidos por estas fontes, torna-se difícil a extração de informações relevantes para estes sistemas.

Por isso é de grande importância a utilização de uma infraestrutura de suporte que proveja serviços comuns a este tipo de aplicação, com o objetivo de minimizar os custos e reduzir o tempo necessário para seu desenvolvimento. A infraestrutura utilizada deve prover serviços para lidar com a natureza dinâmica das aplicações cientes de contexto, onde dispositivos podem entrar ou sair do ambiente ao longo de sua execução.

Uma infraestrutura de suporte para aplicações cientes de contexto, composta de serviços, tais como os serviços de descoberta e contexto, descritos em (SZTAJNBERG, 2009), (SCHMALENSTROEER, 2007) e (LEE, 2006), oferece facilidades para a descoberta e posterior referência aos recursos do ambiente utilizando informações de seu tipo, localização e informações de suas propriedades. Estes serviços permitem ao programador selecionar uma classe de recursos e ainda restringir as opções desejadas com base em informações de contexto. Em soluções como estas, as informações de contexto são fornecidas geralmente como pares <chave-valor> e as interfaces de busca permitem apenas composições simples.

Serviços de descoberta que baseam-se em chaves, não capturam completamente a semântica da consulta recebida, por não considerarem as possíveis relações semânticas entre estas chaves. Nestas soluções, recursos do mesmo tipo, descritos de forma ligeiramente diferente, são tratados como recursos distintos, diminuindo as chances de uma aplicação encontrar os recursos desejados, apesar destes estarem disponíveis. Adicionalmente, em um caso inverso, recursos com descrições sintaticamente equivalentes mesmo que semanticamente diferentes (com funções realmente diferentes) são retornados, causando baixa precisão na busca.

Esta abordagem pode resultar então em baixa taxa de recuperação de recursos e baixa precisão, devido à falta de expressividade semântica, e deixa para o programador a tarefa de identificar e especificar precisamente qual recurso precisa utilizar, para que possa acessá-lo.

Para sanar o problema de falta de expressividade semântica, uma outra abordagem, baseada em ontologias, pode ser utilizada pelos serviços de descoberta.

Em uma aplicação ciente de contexto, onde múltiplas variáveis, contendo informações de diferentes domínios e potencialmente conflitantes, são manipuladas, os componentes da aplicação precisam compartilhar, utilizar e interpretar de forma inequívoca um mesmo

conjunto de termos. Ontologias provêem um mecanismo estruturado para a representação e compartilhamento de tais informações. Sua utilização pela infraestrutura disponibilizada para sistemas cientes de contexto pode então ser útil para permitir a expressão de objetivos ou interesses em um nível mais alto de abstração durante as consultas.

Os benefícios da interoperabilidade semântica são inúmeros (HEFLIN, 2000). Como descrito, uma consulta pode ser frustrante para um cliente de um serviço de registro e descoberta devido à limitação das técnicas de busca baseadas em palavras-chave. Tais consultas podem retornar erroneamente sem resultados ou com muitos resultados irrelevantes. A razão disso é a ausência de semântica no registro dos recursos (SONG, 2005).

Ao acrescentar semântica à descrição dos recursos, se existirem recursos disponíveis do mesmo tipo, com descrições diferentes, estes serão semanticamente interligados e podem eventualmente ser usados pela aplicação. Nesta abordagem, ontologias são utilizadas para a classificação dos serviços com base em suas propriedades. Isto permite que a recuperação de serviços seja baseada em suas características e não apenas em sua denominação (identificação).

Ontologias têm sido estudadas e desenvolvidas na área de inteligência artificial para facilitar o compartilhamento e reutilização de conhecimento (BROENS, 2004). Uma vantagem obtida com o uso de ontologias é a prevenção de diferentes interpretações a respeito da semântica dos termos de um domínio.

Ontologias têm sido utilizadas para resolver problemas que exigem intercâmbio de informações estruturadas em diversas áreas da ciência da computação tais como: representação de conhecimento, processamento de linguagem natural e modelagem conceitual, entre outras. A abordagem declarativa utilizada pelas ontologias permite descrever um domínio de conhecimento sem qualquer compromisso com a implementação de um sistema de *software*, ou seja, o conhecimento modelado é independente de plataforma de *software* e *hardware*.

Neste trabalho, será apresentada uma abordagem que tem como objetivo especificar uma camada de *software* capaz de acrescentar informações semânticas às descrições de recursos disponibilizados para aplicações cientes de contexto. Isto se dará através da modelagem das informações sobre os recursos em ontologias. O objetivo é ter um processo de descoberta de recursos para a autoadaptação de sistemas sensíveis a contexto mais preciso e flexível.

Como contribuição deste projeto, a atividade de descoberta de recursos e a obtenção de informações de contexto em aplicações cientes de contexto, poderão ser realizadas utilizando informações de mais alto nível, com maior expressividade, e utilizando termos do domínio da aplicação, provendo assim ferramentas melhores para a adaptação dinâmica, dadas as necessidades de tais sistemas. Este projeto oferece também pontos de extensão para que as aplicações clientes definam e utilizem suas próprias ontologias para a obtenção de informações de contexto.

A abordagem proposta foi integrada ao projeto descrito em (RODRIGUES, 2009a), acrescentando semântica aos serviços propostos. A reestruturação destes serviços permite que o usuário encontre recursos a partir da análise semântica dos termos pesquisados, retornando assim recursos que não seriam encontrados anteriormente.

São também propostas, neste trabalho, aplicações de exemplo que demonstram as possibilidades de utilização dos serviços de descoberta e contexto disponibilizados, e as melhorias obtidas com a utilização desta abordagem baseada em ontologias. A infraestrutura desenvolvida foi avaliada em termos de desempenho e uma comparação com soluções correlatas também foi realizada, apresentando como esta proposta se destaca em relação a outras.

Organização do texto

O trabalho está estruturado da seguinte forma: no Capítulo 1 são apresentados os conceitos básicos relacionados à nomeação, representação, descoberta e monitoração de recursos. São apresentados o conceito de ontologias (adotadas neste trabalho como solução de representação de recursos) e a infraestrutura que serviu como base para o desenvolvimento deste trabalho. No Capítulo 2 é apresentada a modelagem utilizada para a representação dos recursos e os serviços disponibilizados para acesso às informações sobre os recursos. A Ontologia de Recursos é especificada no Capítulo 3. São detalhadas as informações representadas através da ontologia. São apresentados também os pontos de extensão disponibilizados e como a Ontologia de Recursos pode ser estendida através da definição de regras de inferência. O Capítulo 4 descreve a implementação de referência do OCDRF. São apresentados também os resultados da análise de desempenho realizada. No Capítulo 5 são apresentadas as aplicações de exemplo definidas para apresentar a utilização e flexibilidade provida pelo OCDRF, pela Ontologia de Recursos e por seus pontos de extensão. No Capítulo

6, são analisados trabalhos relacionados. São destacados os pontos fortes e fracos de cada trabalho e é feita uma comparação entre estes trabalhos, demonstrando o diferencial da presente proposta. Finalmente, no Capítulo 7 são apresentadas as conclusões e trabalhos futuros.

1 CONCEITOS BÁSICOS

Este capítulo aborda conceitos básicos relacionados à nomeação, representação, descoberta e monitoração de recursos. São descritas formas de uma aplicação ciente de contexto expressar buscas por recursos do seu contexto de execução, e como estas consultas podem ser melhoradas através de análises semânticas. É apresentado o conceito de ontologia, e como a introdução de análises semânticas pode prover serviços de descoberta e contexto com maior taxa de recuperação e maior precisão em comparação aos serviços baseados em pares <chave-valor>. Também é apresentada a infraestrutura que serviu como base para o desenvolvimento deste trabalho.

1.1 Descoberta de recursos

Um dos principais requisitos para o funcionamento de sistemas cientes de contexto é um mecanismo de descoberta de recursos. Uma infraestrutura de suporte disponibilizada para este tipo de sistema deve fornecer uma abordagem de casamento (*matching*), que correlacione parâmetros de busca com informações do contexto. Este requisito é essencial para permitir que a aplicação cliente tenha acesso aos recursos necessários, de acordo com seu contexto de execução.

Outro ponto central da computação ciente de contexto é a questão de como tratar novos elementos introduzidos dinamicamente no contexto de execução das aplicações (YE, 2007). Estes novos elementos precisam ser descritos e registrados de forma que possam ser localizados posteriormente. Logo, estes elementos devem ser submetidos a um processo de registro para que se integrem ao contexto de execução da aplicação. A partir do processo de registro, estes recursos passam a estar acessíveis através do processo de descoberta para a utilização por parte das aplicações clientes.

A tecnologia de buscas tradicional utilizada pelos serviços de descoberta é baseada no processamento da igualdade entre palavras-chave, entre a descrição dos recursos e os termos das buscas realizadas pelas aplicações clientes. Porém, esta forma de busca apresenta restrições na utilização dos termos e pode não fornecer o resultado esperado. As mesmas palavras-chave podem ter significados diferentes em contextos diferentes, enquanto, por vezes, palavras-chave diferentes podem ter significados semelhantes. Essa dificuldade em

compreender o significado semântico das palavras-chave pode resultar em desempenho insatisfatório nas buscas. Ontologias, descritas na próxima seção, reforçam os mecanismos de busca, que passam a ter capacidade de se referir a um conceito preciso ao invés de a uma simples palavra-chave.

A introdução de semântica, através de ontologias, nos serviços de descoberta disponibilizados para aplicações sensíveis a contexto, pode levar à criação de uma nova geração de serviços, baseados no significado e não apenas na sintaxe de termos. Assim, as pesquisas irão recuperar referências a recursos relacionados com a semântica da solicitação do usuário (BONINO, 2004).

1.2 **Ontologia**

O termo Ontologia utilizado originalmente na filosofia, vem do grego *ontos*, “ser”, “ente”; e logos, “saber”, “doutrina”. Significando em sentido estrito, o “estudo do Ser” (PORTAL DO ESTUDANTE DE FILOSOFIA, 2011).

No dicionário (WEBSTER, 2011), uma ontologia é definida como: “um ramo da Metafísica preocupado com a natureza e as relações do Ser, ou uma teoria particular sobre a natureza do Ser ou os tipos de existências”.

A essência da ontologia é capturar as características naturais de realidades e relações entre realidades (YE, 2007).

O termo foi adotado pelas comunidades de Inteligência Artificial e Gestão do Conhecimento para se referir a conceitos e termos usados para descrever alguma área do conhecimento ou construir uma representação deste.

Considerando um domínio como uma área de estudo ou área de conhecimento específica, como medicina ou gestão financeira, uma ontologia define os termos utilizados para descrever e representar um domínio (W3C1, 2004).

Mais formalmente, uma ontologia especifica uma conceitualização acerca de um domínio. O termo conceitualização corresponde a uma coleção de objetos, conceitos e outras entidades existentes em um domínio e os relacionamentos entre eles. Uma conceitualização é uma visão abstrata e simplificada do mundo que se deseja representar. Para isto são definidos axiomas que restrinjem as possíveis interpretações dos termos definidos (GRUBER, 1995).

As ontologias capturam e especificam de forma geral, formal e explícita o conhecimento de um domínio com sua semântica intrínseca através de terminologias consensuais, axiomas formais e restrições. Ou seja, as ontologias devem envolver um vocabulário bem formado com relações claramente definidas entre os diferentes termos (YE, 2007).

Nas ontologias, a semântica é concretizada por um conjunto de termos, relações entre termos e regras de inferência. Este é o núcleo das ontologias. Os termos constituem um vocabulário controlado com definições explícitas. As relações entre termos incluem as relações entre as instâncias, entre as classes e suas instâncias, e entre as classes. Enquanto as regras de inferência permitem definir conhecimento sobre um sistema, que poderia estar oculto ou implícito, e permitem derivar novos conhecimentos a partir de fatos existentes (YE, 2007).

O vocabulário comum e a especificação precisa da semântica oferecida pelas ontologias abrem caminho para o processamento automático de informações, uma vez que estas informações semânticas podem ser também entendidas por máquinas e não somente por seres humanos (BROENS, 2004).

O objetivo do desenvolvimento de ontologias na área de sistemas distribuídos é a criação de artefatos que diferentes aplicações possam compartilhar e usar para interagir umas com as outras. As ontologias fornecem um mecanismo bem fundamentado para a representação e troca de informações estruturadas. Podem então ser usadas para a comunicação entre os módulos de um sistema ou para a interação entre sistemas distintos.

1.2.1 Aplicações na área de Informática

O termo “ontologia” foi emprestado da filosofia e introduzido no campo da engenharia do conhecimento como forma de abstrair e representar conhecimento. Ontologias têm sido utilizadas em ferramentas de busca conceitual ou semântica, em agentes de *software*, sistemas de apoio à decisão, sistemas de compreensão de fala e linguagem natural, gestão de conhecimento, bases de dados inteligentes, comércio eletrônico e principalmente no desenvolvimento da Web Semântica.

Ontologias têm sido utilizadas em aplicações na área de informática por fornecerem terminologias padronizadas e semântica rica, facilitando o compartilhamento e a reutilização

de informações. As ontologias permitem o acesso efetivo e eficiente a fontes de informação heterogêneas e distribuídas.

Interoperabilidade é um ponto chave para a integração de sistemas de computação heterogêneos e as ontologias têm sido amplamente utilizadas para resolver a heterogeneidade sintática com o objetivo de integrar sistemas e bases de dados.

Através do uso de ontologias, são evitadas diferentes interpretações a respeito da semântica dos termos de um domínio. Sistemas compartilham as mesmas terminologias, o entendimento de classes, propriedades e indivíduos, e como estes elementos se relacionam entre si.

A utilização de ontologias permite a integração transparente de serviços. Os conceitos definidos em uma ontologia podem ser publicamente compartilhados através da Internet, cabendo aos desenvolvedores de tais sistemas saberem a localização de cada ontologia necessária para utilizá-las (BULCÃO, 2006).

A abordagem declarativa utilizada pelas ontologias permite descrever um domínio sem qualquer compromisso com a implementação de um sistema de *software*, ou seja, o conhecimento modelado é totalmente independente de implementação.

Outra vantagem obtida com a utilização de ontologias é a capacidade de reutilizar e estender definições de termos de outras ontologias (BULCÃO, 2006). Esta tarefa é realizada através da importação de outras ontologias conhecidas.

1.2.1.1 Web Semântica

A área onde as ontologias mais têm se destacado é na Web Semântica. As ontologias podem ser utilizadas para melhorar as aplicações Web atuais e podem permitir novas utilizações da Internet, através de anotações semânticas em *Web Services*. As ontologias participam da emergente Web Semântica como a forma de representar a semântica de documentos, possibilitando que a semântica seja utilizada por aplicações web e agentes inteligentes.

Não há dúvida que a Web é um imenso sucesso, ajudando desde estudantes pré-escolares a CEOs a compartilhar informações, simplificar pesquisas e conduzir negócios em tempo real. Ainda assim, muitos pesquisadores não estão felizes com o estado atual da tecnologia Web. As páginas (a linguagem) HTML foram desenvolvidas para serem lidas por

humanos, não por máquinas. Compradores via Internet sabem que o código (ou nome) de um item apresentado em uma página Web e o preço ao lado do mesmo são referentes à imagem apresentada logo a seguir. Porém uma máquina não conseguiria fazer a mesma inferência, já que HTML não foi criado para conter este nível de significado. Em HTML podemos expressar: “Coloque esta imagem próxima deste texto”, mas não há mecanismo semântico para expressar que estes pedaços de informação possuem sentido especial juntos, que estes preços se referem a estes itens, por exemplo (BAUM, 2006).

Os *hiperlinks* hoje incluem pouca informação sobre o relacionamento entre os *sites* ou entre os elementos em uma página. Esta limitação restringe a Web de atingir seu potencial como meio no qual informações podem ser compartilhadas e processadas por ferramentas automatizadas, assim como pelas pessoas. Se documentos Web forem definidos semanticamente, pesquisas simples poderão identificar facilmente as informações desejadas. A Web Semântica é uma forma proposta para mapear conjuntos de informações, em princípio não correlatas, baseada em critérios definidos por especialistas (BAUM, 2006). Ela é uma visão para o futuro da Web na qual a informação possuirá significado explícito, tornando mais fácil para as máquinas processar e integrar automaticamente informações disponíveis na Web.

Um elemento necessário para a Web Semântica é uma linguagem formal de ontologia que possa descrever a semântica das classes e propriedades utilizadas nos documentos web. Esta tarefa é alcançada através da utilização de padrões como o RDF (*Resource Description Framework*) e OWL (*Web Ontology Language*) apresentados na Seção 1.2.6. Estes padrões e descritores permitem que desenvolvedores Web adicionem significado a documentos Web, fornecendo uma infraestrutura para a definição de como as informações serão conectadas e como as relações desejadas serão expressas.

Assim como ocorre quando metadados são adicionados a uma base de dados, a Web Semântica inclui informações sobre os relacionamentos entre os elementos. Estes elementos são definidos em sentenças autodescritivas chamadas triplas RDF contendo um substantivo, predicado e objeto. Modelando as informações desta forma, desenvolvedores podem construir relacionamentos entre os elementos que eram anteriormente tratados como entidades separadas (BAUM, 2006).

Buscas inteligentes na Internet são provavelmente o melhor exemplo da tecnologia da Web Semântica em ação. Em buscas na Web típicas, a *string* exata procurada precisa estar presente em todos os documentos resultantes. Sinônimos e expressões de linguagem que representem o mesmo conceito não são levados em consideração. Por exemplo, uma busca

por “música” não trará resultados que contenham “flauta” a não ser que a página também contenha a *string* “música”. Se os serviços de busca da Web fossem capazes de compreender o contexto de “música” e os elementos relacionados ao termo, eles seriam capazes de saber que termos como “música” e “flauta” estão interligados, mesmo que não sejam explicitamente mencionados no critério de busca.

As características da Web Semântica satisfazem a um nicho de indústrias que trabalham intensivamente com informações, como a farmacêutica e as forças armadas, onde toda técnica capaz de acelerar o tempo no qual uma informação é alcançada é extremamente valorizada (BAUM, 2006).

1.2.2 Características de uma ontologia

Definir termos em uma ontologia não é muito diferente do que desenvolvedores fazem para uma base de dados, por exemplo. A mesma lógica embutida em um diagrama Entidade-Relacionamento permite ao desenvolvedor distinguir entre objetos e relacionamentos em uma ontologia (BAUM, 2006).

Algumas questões precisam ser abordadas na elaboração de ontologias para um domínio: (i) a determinação do escopo e finalidade das ontologias; (ii) a captura dos termos apropriados; (iii) a codificação das ontologias com linguagens formais; (iv) a avaliação iterativa no processo de construção; (v) a avaliação iterativa das próprias ontologias em termos de clareza, coerência, compromisso ontológico, ortogonalidade, influência da tecnologia utilizada e extensibilidade e (vi) a documentação formal das ontologias criadas (YE, 2007).

Os termos da ontologia devem ser definidos com clareza; i.e., devem ser descritas condições necessárias e suficientes para que eles possam ser identificados de forma não ambígua. As definições precisam ser consistentes e coerentes. Ou seja, os termos não podem ser utilizados de duas maneiras diferentes em uma mesma ontologia.

As ontologias devem descrever apenas os conceitos do domínio que estiverem no seu escopo para o compartilhamento de conhecimento. Elas devem se ater ao compromisso ontológico firmado, sob pena de limitar sua extensibilidade. Conceitos genéricos devem ser definidos de forma independente e desacoplada, para garantir sua ortogonalidade. Deve-se evitar que as escolhas para representação sejam realizadas pela conveniência da notação ou

implementação. As ontologias devem ser extensíveis, de forma que seja fácil adicionar novos termos sem causar ambiguidade ou confusão com a definição de termos existentes. Além disso, as ontologias devem ser documentadas após sua finalização. Todas as decisões e premissas usadas na definição dos termos devem ser documentadas para facilitar o compartilhamento das ontologias criadas.

1.2.3 Tipos de Ontologias

Diferentes tipos de ontologias são construídas para tipos diferentes de aplicações, e possuem níveis diferentes de detalhes. As classificações das ontologias são resumidas em (YE, 2007) em termos de generalidade como a seguir.

- Ontologias genéricas: são aquelas que descrevem conceitos gerais, independentes de qualquer domínio particular (exemplo: tempo, processos);
- Ontologias de domínio: são aquelas que descrevem os conceitos de um domínio em particular (como Biologia, Física ou Direito Tributário); e
- Ontologias de aplicação: são aquelas que descrevem os conceitos necessários para aplicações específicas e podem ser construídas sobre ontologias genéricas e de domínio (como ontologias para auxiliar na identificação de doenças coronarianas).

1.2.4 Elementos de uma Ontologia

Cada ontologia define um conjunto de classes, relações, funções e constantes para algum domínio do discurso e inclui axiomas para restringir sua interpretação (GRUBER, 1995). Em mais detalhes, os componentes básicos de uma ontologia são: (i) classes, (ii) propriedades e (iii) instâncias.

Classe. Uma classe representa um conjunto de características que elementos (indivíduos) podem possuir. Elas agrupam as definições de objetos comuns e são organizadas hierarquicamente.

Propriedade. Uma propriedade é uma relação binária, i.e., relaciona os elementos de um domínio com elementos em um intervalo (*range*) definido. Elas são usadas para estabelecer relações entre indivíduos ou entre indivíduos e valores estáticos. Existem dois tipos de

propriedades: *datatype properties* (onde o *range* é um literal RDF ou um tipo definido em um XML *Schema*) e *object properties* (onde o *range* é uma classe da ontologia) (W3C2, 2004).

Instância. Uma instância representa um indivíduo específico de uma classe, ou seja, as instâncias representam os dados do domínio.

Para expressar os variados tipos de relacionamentos que podem existir entre os elementos básicos de uma ontologia, são estabelecidos relacionamentos semânticos. Tais relacionamentos fornecem um mecanismo poderoso para o raciocínio (inferência) sobre as propriedades e sobre os elementos que elas relacionam. Através da utilização dos relacionamentos semânticos é possível expressar relações como as utilizadas em modelagens Orientadas a Objetos como: especialização, generalização e composição, além de outras que não podem ser expressas utilizando modelagem Orientada a Objeto como: equivalência, disjunção, simetria e transitividade.

A seguir, são descritos alguns tipos de relacionamentos que podem ser atribuídos às propriedades de uma ontologia (W3C2, 2004):

- Simetria

  Uma propriedade P simétrica define que para todo x e y, se P(x,y) então P(y,x). Como exemplo, podemos ter a relação *casadoCom* (com domínio e *range* da classe Pessoa).

- Transitividade

  Uma propriedade P transitiva define que para todo x, y e z, se P(x,y) e P(y,z) então P(x,z). Como exemplo, podemos ter a relação *maisAltoQue* (com domínio e *range* da classe Pessoa).

- Propriedade Funcional

  Uma propriedade P funcional define que um elemento do domínio pode ter apenas um elemento associado a ele no intervalo (*range*) definido. Então para todo x, y e z, se P(x,y) e P(x,z) então y=z. Como exemplo, podemos ter a relação *possuiIdade* (domínio: Pessoa, *range*: *number*).

- Propriedade Inversa

  Uma propriedade P1 inversa a P2 define que para todo x e y, se P1(x,y) então P2(y,x). Como exemplo, podemos ter a relação *maeDe* e *FilhoDe* (com domínio e *range* da classe Pessoa).

- Propriedade Inversamente Funcional

  Uma propriedade P é inversamente funcional se seu inverso for funcional. Uma propriedade P inversamente funcional define que para todo x, y e z, se P(y,x) e P(z,x) então y=z.

As ontologias ainda permitem a definição de restrições sobre as propriedades e regras de inferência que podem ser avaliadas utilizando mecanismos de inferência. Motores de inferência são utilizados para verificar a consistência e a integridade das ontologias e deduzir novos conhecimentos com base nas regras ou nas relações dos conceitos especificados nas ontologias (YE, 2007). Com base nas regras de inferência e nas restrições estabelecidas sobre as propriedades de cada classe, uma máquina de inferência é capaz de deduzir de forma automática informações não contidas explicitamente nas ontologias.

Máquinas de inferência podem ser utilizadas para habilitar responsáveis por tomadas de decisão a procurar e resgatar informações a partir de fontes de dados dispersas. Por exemplo, se militares encontrarem um *container* em um veículo abandonado, eles podem fazer uma busca que conecte os elementos químicos contidos no *container* com o fabricante do mesmo, o tipo de veículo e a sua localização utilizando diversas ontologias (BAUM, 2006).

1.2.5 Passos para a Construção de uma Ontologia

A seguir são enumeradas fases importantes do processo de construção de ontologias (BULCÃO, 2006):

1. Decidir a que domínio a ontologia deve atender, sua finalidade e quais questões a ontologia deve ser capaz de responder.
2. Considerar a utilização de ontologias pré-existentes.

3. Enumerar os termos importantes do domínio. Neste passo não deve haver preocupação com relacionamentos entre os termos, com os atributos que os conceitos podem ter ou se os conceitos são classes ou atributos.
4. Definir a estrutura/hierarquia de classes. É importante não confundir nomes de um conceito com o conceito em si durante a definição de classes.
5. Definir as relações entre as classes.
6. Definir os atributos das classes para que se possa responder às questões definidas na delimitação do domínio e no escopo da ontologia. Os atributos das classes podem também ser obtidos dentre a lista de termos criada anteriormente.
7. Definir as restrições sobre os atributos, como: o tipo do valor armazenado e a sua cardinalidade.
8. Criar as instâncias.

1.2.6 Linguagens Utilizadas para a Construção de Ontologias

As linguagens de ontologia formalmente especificadas tornam a semântica explícita e inequívoca para que as ontologias sejam passíveis de tratamento e integração automatizados (YE, 2007). A linguagem utilizada neste trabalho para a codificação de ontologias foi o OWL, atualmente denominado OWL 1.

1.2.6.1 OWL

OWL (*Web Ontology Language*) (W3C2, 2004) é a linguagem recomendada pelo W3C (W3C3, 2011), baseada em RDF (*Resource Description Framework*) (W3C6, 2004) para a criação de ontologias. A seguir são descritas as tecnologias referentes à codificação de ontologias e a relação existente entre elas. Estas tecnologias são amplamente utilizadas no campo da engenharia de ontologias e, especificamente, em aplicações de computação sensível ao contexto.

A base da codificação OWL é o XML (*eXtensible Markup Language*), que é uma linguagem padronizada para descrever dados de forma estruturada. Em XML, os dados são rotulados com elementos definidos pelo usuário. Para restringir as estruturas sintáticas de

documentos XML, foram criados o DTD (*Document Type Definition*) e posteriormente o XMLS (XML *Schema*).

Os DTDs e XML Schemas são suficientes para a troca de dados entre partes que concordaram com uma definição *a priori*, porém a sua falta de semântica impede que máquinas realizem essa tarefa com novos vocabulários de forma confiável. Observando que um termo pode ser usado com significados diferentes em contextos diferentes e diferentes termos podem ser utilizados com o mesmo significado.

O RDF e o RDFS (RDF *Schema*) começaram a abordar este problema, impondo limitações estruturais ao XML e permitindo que uma semântica simples fosse associada aos identificadores. O RDF fornece uma maneira inequívoca e extensível de expressar declarações simples no formato "sujeito, predicado, objeto", onde o sujeito representa um recurso como uma URI (*Uniform Resource Identifier*), o objeto pode ser um literal ou uma referência para a URI de outro recurso, e um predicado é uma relação entre o sujeito e o objeto (YE, 2007).

O RDF Schema é uma linguagem simples de ontologia que estende o RDF. Com o RDFS, pode-se definir classes que podem ter várias subclasses e super classes e pode-se definir propriedades, que podem ter subpropriedades, domínios e intervalos. Porém, para atingir interoperabilidade entre inúmeros esquemas desenvolvidos e geridos autonomamente, é necessária uma semântica mais rica do que a provida pelo RDFS. O RDFS não consegue especificar que as classes "Pessoa" e "Carro" são disjuntas ou que um "quarteto de cordas" tem exatamente quatro músicos como membros, por exemplo.

Sobre o XML(S) e o RDF(S) foram criados o OIL (*Ontology Interchange Language*) e o DAML (DARPA *Agent Markup Language*); projetados para suportar uma semântica formal e raciocínio eficiente. Com a junção de DAML e OIL, DAML+OIL foi formado, e foi proposto como um padrão W3C para a representação ontológica e de metadados. DAML+OIL (HARMELEN, 2001) suporta estruturas primitivas e estruturas de dados mais complexas, semânticas bem estruturadas e procedimentos de inferência baseados em lógica de descrição (YE, 2007). Lógica de descrição é um campo de pesquisa que estuda um fragmento da lógica de primeira ordem (W3C2, 2004).

O OWL foi criado com base em DAML+OIL. Ele fornece maior facilidade de interpretação dos dados, fornecendo vocabulários adicionais com uma semântica formal (YE, 2007). Ele acrescenta mais vocabulário para descrever propriedades e classes, descreve as

relações entre as classes (por exemplo: classes disjuntas), cardinalidade (por exemplo: “exatamente um”), igualdade, propriedades com tipagem mais rica, características das propriedades (por exemplo: simetria) e classes enumeradas.

Outras linguagens foram usadas anteriormente para desenvolver ferramentas e ontologias para comunidades específicas (especialmente em aplicações de *e-commerce* e em aplicações acadêmicas), porém elas não foram definidas para serem compatíveis com a arquitetura da *World Wide Web* em geral e com a Web Semântica em particular.

O OWL possui:

- Capacidade de ser distribuído em vários sistemas;
- Escalabilidade para as necessidades da Web;
- Compatibilidade com as normas de acessibilidade da Web e internacionalização;
- Transparência e extensibilidade.

Foram desenvolvidas três sublinguagens de OWL, cada uma com crescente grau de expressividade destinada a comunidades específicas de usuários e implementadores: OWL Lite, OWL DL e OWL Full (W3C2, 2004).

- OWL Lite suporta classificação hierárquica e restrições simples. Embora suporte restrições de cardinalidade, apenas permite valores de cardinalidade igual a 0 ou 1. OWL Lite oferece um caminho para a migração rápida de outras taxonomias.
- OWL DL suporta o máximo de expressividade, mantendo completude computacional (todas as conclusões são garantidas de serem computadas) e decidibilidade (todas as computações irão terminar em um tempo finito). Inclui todas as construções da linguagem OWL, mas elas podem ser usadas apenas sob certas condições (por exemplo, enquanto uma classe pode ser uma subclasse de várias classes, uma classe não pode ser uma instância de uma outra classe). OWL DL recebeu este nome devido a sua correspondência com *description logics* (lógica de descrição).
- OWL Full permite máxima expressividade e a liberdade sintática do RDF sem garantias computacionais. Ao usar OWL Full, uma classe pode ser tratada

simultaneamente como uma coleção de indivíduos e como um indivíduo. OWL Full permite a uma ontologia aumentar o significado de vocabulários pré-definidos (em RDF ou OWL).

A escolha de qual sublinguagem OWL usar depende da exigência de expressividade e da complexidade das ontologias alvo (YE, 2007). A escolha entre OWL Lite e OWL DL depende da medida em que são exigidas construções mais expressivas como as fornecidas pelo OWL DL e OWL Full. Adicionalmente, a escolha entre OWL DL e OWL Full depende da medida em que são exigidas facilidades de metamodelagem de RDF Schema (por exemplo, definir classes de classes, ou anexar propriedades às classes).

Para este trabalho, uma vez que serão desenvolvidas ontologias que possuirão relações mais complexas que as utilizadas em taxonomias, a sublinguagem OWL Lite não será utilizada. Como não serão necessárias facilidades de metamodelagem e por ser OWL DL a linguagem mais utilizada nos trabalhos pesquisados, foi esta então selecionada como sublinguagem OWL para ser utilizada.

1.2.6.2 Editores de ontologia

Neste trabalho não foram utilizados editores específicos para a criação das ontologias. Porém editores de ontologia, como o Protégé (NOY, 2000), podem ser usados pelo administrador do ambiente ciente de contexto para editar as ontologias criadas neste trabalho.

1.3 **CDRF**

Em (RODRIGUES, 2009a), foi proposta a utilização de um modelo para a descoberta e monitoração de recursos com base em dados de contexto. A arquitetura criada para o CDRF (*Contextual Discovery of Resources Framework*) apresenta os seguintes elementos: Agentes de Recursos (AR), um Serviço de Registro e Diretório (SRD), um Serviço de Contexto (SC) e um Serviço de Descoberta (SD). Toda comunicação entre os elementos da arquitetura ocorre seguindo *schemas* XML especificados para as consultas e respostas de cada elemento.

A Figura 1 ilustra os componentes da infraestrutura base para o OCDRF, proposto no presente trabalho, apresentando os relacionamentos existentes entre estes componentes.

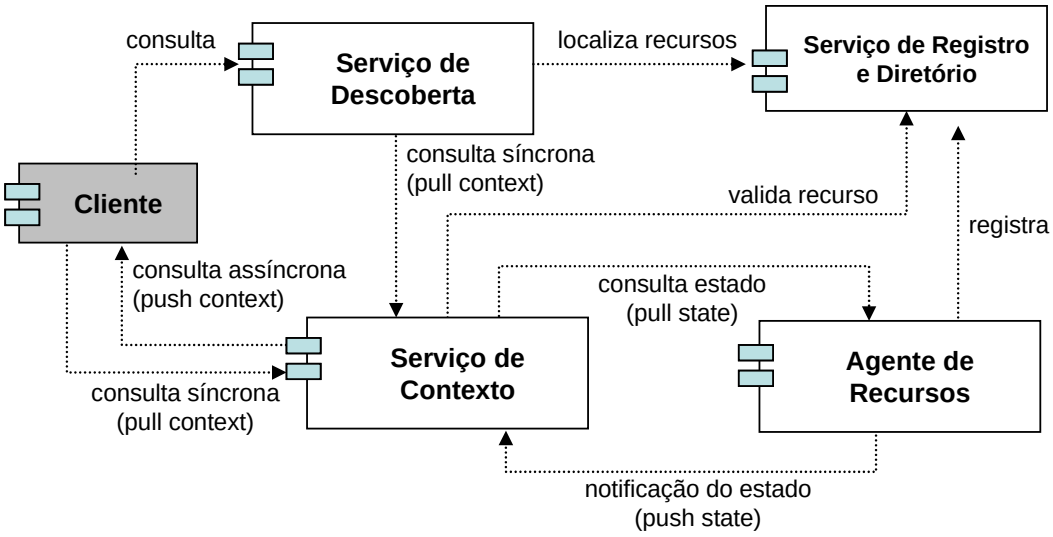

Figura 1 - Componentes da infraestrutura base para o OCDRF (RODRIGUES, 2009a)

Os Agentes de Recursos (ARs) representam os recursos disponíveis para as aplicações cientes de contexto. São os elementos da infraestrutura responsáveis por coletar as diversas medidas dos sensores ou dos componentes de *software* presentes no contexto de execução das aplicações. Eles escondem os detalhes de baixo nível utilizados no sensoriamento e aquisição dos dados brutos. A especificação dos ARs descreve uma interface uniforme para que a aplicação não tenha que estar ciente dos detalhes específicos de cada tipo de mecanismo de sensoriamento.

Todo AR possui associado a ele uma descrição em XML especificando o recurso que ele representa. Esta descrição em XML deve estar em conformidade com o XML *Schema* especificado para o registro dos recursos. Na descrição de um AR são informadas suas características e dados operacionais, como sua localização e mecanismos de acesso suportados. Esta descrição deve ser registrada no repositório central quando o AR for iniciado.

Os ARs podem possuir atributos estáticos e atributos que sofrem variações ao longo do tempo (atributos dinâmicos). O Código 1 apresenta uma mensagem de registro de um AR com tipo "*BloodPressure*", descrevendo seus atributos (Linhas 3-8) e informações operacionais (Linhas 9-10).

```
1  <ResourceRegister>
2    <Type>BloodPressure</Type>
3    <Attributes>
4      <Attribute Name="serial" Value="TelcoMed004E"/>
```

```
 5       <Attribute Name="model"   Value="WristClinic"/>
 6       <Attribute Name="systolic" Value="Dynamic"/>
 7       <Attribute Name="diastolic" Value="Dynamic"/>
 8    </Attributes>
 9    <URI>hhs/BPAgent</URI>
10    <IP>152.92.155.196</IP>
11  </ResourceRegister>
```

Código 1 - Exemplo de uma mensagem de registro de um AR no CDRF (RODRIGUES, 2009a)

Um AR está sempre associado a um tipo de recurso, conforme apresentado no Código 1 (Linha 2). Um tipo de recurso representa uma categoria de recursos que pode estar presente no contexto de execução das aplicações. Assim como os próprios recursos, as descrições dos tipos de recursos também são registradas no repositório central, a partir do qual as mesmas podem ser recuperadas para posterior utilização. Elas funcionam como um *template* e como um contrato que todos os recursos de um determinado tipo devem seguir. Nesta infraestrutura os tipos de recursos precisam ser registrados manualmente pelo administrador do sistema no SRD.

O Serviço de Registro e Diretório (SRD) faz o papel de repositório central na arquitetura e serve como ponto a partir do qual os recursos e os próprios componentes da infraestrutura podem ser localizados. O Serviço de Contexto e o Serviço de Descoberta também se registram junto ao SRD e podem ser considerados como recursos especiais da infraestrutura. O SRD armazena as propriedades estáticas dos recursos e permite que consultas sejam efetuadas sobre elas.

Os clientes da infraestrutura podem ter acesso aos dados de contexto disponibilizados pelos ARs através do Serviço de Descoberta (SD) ou por meio do Serviço de Contexto (SC). Os ARs ficam então “blindados” pelo SD e SC, do ponto de vista da aplicação.

Quando uma aplicação cliente não conhece a localização de um AR, ela pode efetuar uma consulta ao SD baseada no tipo e atributos do recurso desejado. O SD consulta o SRD e obtém uma relação de recursos que respeitam os critérios passados como parâmetros da consulta. Em seguida, o SD verifica quais propriedades dos recursos identificados sofrem variações de acordo com o contexto (quais propriedades são dinâmicas) e efetua uma consulta ao SC para obter os valores mais atuais destas propriedades. De posse destes valores, o SD é finalmente capaz de determinar quais recursos estão de acordo com os critérios definidos pela aplicação cliente e lhe retorna uma relação contendo estes recursos. A lista de ARs retornada

pelo SD deve ser então classificada pela aplicação cliente de forma a identificar o melhor recurso dentre os que satisfazem as restrições especificadas.

Quando uma aplicação já possui a referência para os recursos a serem monitorados, ela pode consultar o SC diretamente para coletar valores de propriedades estáticas e dinâmicas destes recursos.

Uma consulta ao SC pode conter um ou mais recursos-alvo. Assim que o SC recebe uma consulta, as propriedades de cada recurso são verificadas junto ao SRD e então o SC consulta cada AR envolvido para obter as informações de contexto individuais atualizadas. Após coletar e consolidar todas as informações necessárias, o SC retorna finalmente o resultado para o cliente.

As consultas ao SC podem ser síncronas (*pull context*) ou assíncronas (*push context*) como apresentado na Figura 1. Nas consultas síncronas, os clientes ficam em estado de espera até que o SC retorne os valores desejados. Já nas consultas assíncronas, o cliente efetua seu registro junto ao SC indicando o tipo de recurso ou o AR no qual está interessado e quando ocorrem mudanças nas propriedades dos ARs registrados, o SC se encarrega de informá-las ao cliente. O SC armazena uma lista contendo os observadores registrados e informa a estes sempre que ocorre uma mudança nas propriedades dos ARs que correspondam aos critérios especificados nas mensagens de registro. O SC, por sua vez, também pode efetuar consultas aos ARs de forma síncrona (*pull state*) ou pode ser informado pelos ARs de forma assíncrona (*push state*).

A proposta da presente dissertação utiliza a infraestrutura disponibilizada pelo CDRF como ponto inicial para o desenvolvimento de serviços, baseados em ontologias, para prover capacidade de análise semântica às buscas efetuadas pelas aplicações clientes.

A integração do modelo de ontologias com o CDRF se deu através de modificações na camada de descrição dos Agentes de Recursos e nos Serviços de Descoberta, de Registro e Diretório e de Contexto da infraestrutura. A reestruturação destes serviços permite que o usuário encontre os recursos desejados, incluindo aqueles que não seriam localizados a partir de simples análises sintáticas das consultas.

2 OCDRF

Este capítulo apresenta a modelagem utilizada para a representação dos recursos para que estes sejam disponibilizados para aplicações cientes de contexto, através da infraestrutura proposta. São apresentados também os serviços disponibilizados pela infraestrutura para acesso às informações sobre os recursos, e as novas características destes serviços suportadas por análises semânticas, acrescentadas com a utilização de ontologias.

2.1 Modelo de Recursos

Os dispositivos eletrônicos disponíveis no mercado estão cada vez mais versáteis. A cada nova geração de produtos eletrônicos lançada, uma quantidade maior agrega características e funcionalidades bem distintas em comparação com o propósito para o qual estes produtos foram desenvolvidos originalmente. Os equipamentos modernos, a cada ano que passa, agregam mais funções e comportamentos. Hoje geladeiras possuem filtros e estes gelam água. As funções estão cada vez mais integradas e os equipamentos, cada vez mais contidos uns nos outros. A tendência observada é de diversificação, i.e., cada vez mais produtos possuírem mais de uma função.

Além da tendência de diversificação de funções, pode-se observar a tendência de compartilhamento e agregação de funções. Um equipamento criado com determinada função e amplamente utilizado para executar esta função, pode agora agregar vários novos comportamentos, comportamentos estes que os usuários se acostumaram a ver executados por outros equipamentos. Um telefone celular originalmente criado para a comunicação de voz, hoje “é também” uma máquina fotográfica, um equipamento de som, um videogame e um receptor GPS, por exemplo. Além disso, através da criação de novos aplicativos, somados à conexão à Internet, este equipamento passa a ter infinitas possibilidades de utilização.

Um exemplo de diversidade e agregação de funções de um equipamento é a utilização de um *smartphone* para a realização de testes oftalmológicos. A empresa Valemobi (VALEMOBI, 2011) desenvolveu um *software* voltado especificamente para testes oftalmológicos, denominado *EyePhone* (SABER ELETRÔNICA, 2009). Concebido a partir do projeto de um médico especialista em oftalmologia, o sistema ajuda a realizar exames e

diagnósticos médicos utilizando como ferramenta o *smartphone* da Apple, *iPhone* (APPLE, 2011).

O *EyePhone* auxilia em procedimentos como medição da quantidade de visão e tamanho da pupila, exame da função macular (fundo de olho) e verificação de estrabismo, além do teste da divisão de cores, para detectar se o paciente apresenta daltonismo. Os exames são feitos com base em imagens gravadas no aplicativo, visualizadas pelo paciente na tela de um *iPhone*, para verificar a acuidade da sua visão.

Considerando então este cenário de convergência e diversificação de funções dos equipamentos modernos, passa a ser importante a distinção entre um produto “ser” versus “ter características ou funções de”. Uma geladeira pode ter, por exemplo, um comportamento de monitor por possuir um *display* e apresentar informações visuais para o usuário, além de prover o serviço para o qual foi desenvolvida inicialmente de resfriar alimentos.

Um exemplo de equipamento com funções distintas agregadas, porém que não obteve sucesso de vendas no mercado, é geladeira *Internet Digital DIOS* da LGE (R-S73CT: 730 litros), que pode ser utilizada para navegar na *Internet* e fazer vídeo chamadas, além de armazenar e resfriar alimentos (BROWNE, 2000).

Para mapear as diversas características que um equipamento pode possuir, é proposta neste trabalho, a estruturação dos conceitos que envolvem estes recursos em uma ontologia. Esta ontologia foi denominada Ontologia de Recursos e sua utilização visa prover semântica na descrição dos recursos disponibilizados para a utilização de aplicações sensíveis a contexto. A Ontologia de Recursos foi então incorporada à infraestrutura de suporte, que será descrita nas seções a seguir, de forma que fosse utilizada durante o processo de busca por recursos, disponibilizado por esta infraestrutura.

Na ontologia proposta são descritos o conceito de recurso, seus componentes e as suas características. Os componentes, classificados como sensores e atuadores, estão contidos nos chamados recursos. Eles possuem associados a eles características (capacidades) e são os reais responsáveis pelas funções providas pelos recursos. A ontologia proposta é então responsável por representar as funcionalidades de cada recurso, através do mapeamento de sensores e atuadores e de suas capacidades, tornando estas informações disponíveis para as aplicações cientes de contexto.

A modelagem dos recursos através de ontologias, utilizando a definição de suas diversas características, proporciona maior flexibilidade ao modelo em comparação à

utilização de uma cadeia de heranças contendo os recursos, que considere apenas seu nome ou “tipo”. Realizar a caracterização de recursos através de cadeia de heranças faz com que um recurso possa pertencer apenas a uma cadeia, não comportando a agregação de características conforme descrito anteriormente.

Assim como os sensores que compõem os equipamentos eletrônicos, componentes de *software* também apresentam a possibilidade de fornecer informações que podem ser monitoradas, e assim como os atuadores contidos nos equipamentos, componentes de *software* também são capazes de processar pedidos de operações, oferecendo suas interfaces, suas APIs (*Application Programming Interfaces*), para a interação com outras aplicações. Dessa forma, os recursos (*Resource*) e os componentes destes recursos (*ResourceComponent*) na ontologia a ser detalhada, também poderão ser usados para representar componentes de *software*.

A Figura 2 apresenta a composição dos dispositivos (ou componentes de *software*) em sensores e atuadores.

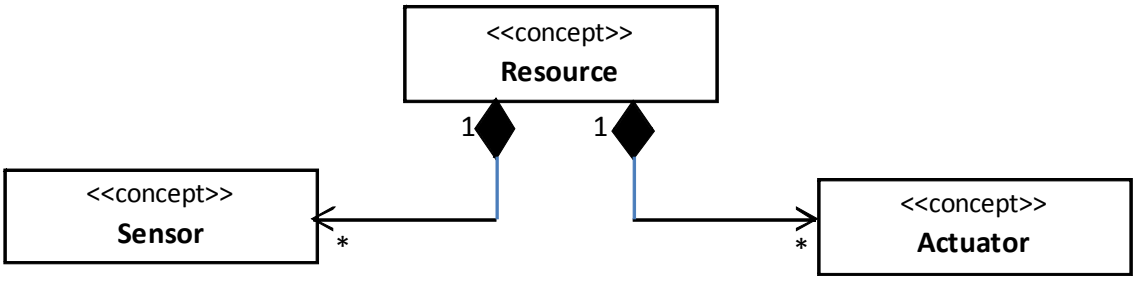

Figura 2 - Componentes gerais da Ontologia de Recursos

A composição apresentada na Figura 2 comporta a representação de um televisor, por exemplo, que possua uma interface através da qual seja possível a uma aplicação verificar o estado deste equipamento, se o mesmo está ligado por exemplo, representada por um sensor, e uma interface através da qual seja possível a uma aplicação enviar mensagens de texto a fim de que sejam apresentadas na tela do televisor, representada por um atuador.

Outro exemplo de equipamento que pode ser utilizado por aplicativos sensíveis a contexto é o SunSPOT (SUNSPOT, 2011). Este equipamento provê diversas funções, que podem ser mapeadas através da Ontologia de Recursos proposta como sensores, são eles: acelerômetros, sensor de luz e sensor de temperatura.

Da mesma forma, uma aplicação pode possuir uma interface através da qual seja possível a outra aplicação monitorar alguma informação do sistema operacional, se uma unidade de disco está atingindo sua capacidade máxima por exemplo, representada por um sensor, e uma interface através da qual seja possível a uma aplicação excluir arquivos antigos de um determinado diretório, representada por um atuador.

Um cenário possível para a utilização do modelo proposto contendo abstrações de recursos, sensores e atuadores em uma aplicação ciente de contexto é uma aplicação para uma residência inteligente, com funções como as descritas a seguir:

A residência de um paciente idoso é monitorada por uma aplicação ciente de contexto capaz de obter dados dos sensores disponíveis neste ambiente. O paciente está em seu quarto e não abre a geladeira por um longo tempo. Isto é observado através de informações de um sensor que indica a abertura da porta, instalado na sua geladeira. O televisor então lhe “diz” que ele deve ir até a geladeira para beber um pouco de água. Em outro cenário da mesma aplicação, esta pode monitorar o sensor da geladeira e “perceber” que a porta foi aberta diversas vezes em um certo intervalo de tempo. A partir desta constatação, a aplicação pode se certificar da disponibilidade de um recurso capaz de medir a glicose do paciente e procurar por um recurso que possua um atuador capaz de enviar informações para o paciente. De posse das informações destes dois recursos, a aplicação pode então enviar para o paciente uma mensagem escrita, de áudio ou vídeo, dependendo da descrição dos dispositivos para envio de avisos encontrados, pedindo-lhe que verifique sua glicose.

A modelagem dos recursos através da Ontologia de Recursos e a infraestrutura de suporte propostos neste trabalho vêm atender aos passos descritos acima para as buscas por sensores e atuadores disponíveis no ambiente e para a interação com os mesmos, servindo como alicerce para o desenvolvimento e funcionamento de aplicações ciente de contexto.

2.2 Serviços de Contexto

Através da utilização da infraestrutura desenvolvida em (RODRIGUES, 2009a), descrita na Seção 1.3, aplicações sensíveis a contexto podem efetuar consultas buscando pelos recursos registrados na infraestrutura. Estas consultas são capazes de realizar o casamento sintático dos termos envolvidos, com base nos nomes dos tipos dos recursos e nos nomes e valores esperados para seus atributos. Estes últimos representando as restrições de contexto que devem ser satisfeitas para que o recurso seja retornado. Como descrito anteriormente, este

tipo de consulta restringe a aplicação, uma vez que esta precisa conhecer exatamente os termos utilizados durante o registro de cada recurso, sob penalidade de não conseguir localizar e utilizar os recursos que lhe sejam necessários.

O Código 2 apresenta um exemplo de consulta que pode ser enviada pelo Serviço de Contexto (SC) ou pelo Serviço de Descoberta (SD) ao Serviço de Registro e Diretório (SRD) da infraestrutura desenvolvida em (RODRIGUES, 2009a), onde um recurso do tipo "*BloodGlucose*" (Linha 2) é buscado.

```
1 <DirectoryQuery>
2    <Type Value="BloodGlucose" Strict="false"/>
3 <DirectoryQuery>
```

Código 2 - Exemplo de consulta sem análise semântica (RODRIGUES, 2009a)

A infraestrutura CDRF também permite que consultas sejam especificadas de forma que todos os subtipos do tipo especificado sejam considerados. Isto é feito através da utilização do atributo "*Strict*" com valor *false* na consulta ao SRD, como apresentado no Código 2 (Linha 2). Para isto, os tipos aceitos para o registro de recursos nos serviços propostos em (RODRIGUES, 2009a) precisam ser previamente cadastrados por um administrador da infraestrutura no Serviço de Registro e Diretório. A partir desta configuração é possível então a uma aplicação localizar os recursos de um determinado tipo utilizando seu nome, como apresentado no Código 2.

Para acrescentar flexibilidade semântica aos serviços oferecidos pelo CDRF, é então proposta no presente trabalho a extensão do modelo de descrição de recursos de forma a empregar ontologias para guiar conversões e casamentos de tipos. Devido ao seu poder de expressividade, o uso de ontologias agrega à infraestrutura a capacidade de efetuar inferências utilizando informações sobre associações entre os recursos do sistema que vão além da simples herança, o que aprimora a capacidade de busca dos serviços providos.

O *framework* desenvolvido foi denominado OCDRF - *Ontology Contextual Discovery of Resources Framework*. Ele possibilita que recursos com diferentes descrições, fornecidos por provedores diferentes, por exemplo, possam ser semanticamente relacionados, ampliando as opções de seleção de recursos de um determinado tipo. Ele também provê dinamismo na inclusão de novos tipos de recursos no sistema, eliminando este trabalho manual por parte dos administradores da infraestrutura.

O OCDRF utiliza uma ontologia denominada neste trabalho como Ontologia de Recursos, para organizar a descrição dos sensores e atuadores e de seus atributos, bem como os relacionamentos existentes entre os mesmos e o seu agrupamento em recursos. Os recursos representam por sua vez, os equipamentos eletrônicos disponíveis para utilização de uma aplicação sensível a contexto ou componentes de *software* disponíveis para o mesmo fim. A ontologia criada é especificada em detalhes no Capítulo 3.

2.3 Características do OCDRF

O OCDRF permite que uma aplicação cliente faça consultas por recursos sem precisar especificar o nome exato de seu tipo e os nomes dos atributos utilizados para o registro destes recursos na infraestrutura. A análise das consultas é realizada com base na semântica dos termos pesquisados e não apenas na sua sintaxe. Isto permite a utilização mais abrangente de termos do contexto de execução da aplicação.

A arquitetura dos serviços providos pelo CDRF foi mantida no OCDRF. Estes serviços foram modificados para que fossem acrescentadas análises semânticas às suas funcionalidades, através da utilização de ontologias.

O OCDRF faz a tradução através da utilização de ontologias (i) das características em termos de tipos de capacidades dos recursos pesquisados, sejam eles equipamentos eletrônicos ou componentes de *software* e (ii) dos atributos que restringem as pesquisas por recursos; buscando por informações que apresentem equivalência semântica.

O objetivo das modificações trazidas pelo OCDRF foi prover às aplicações clientes o máximo de flexibilidade na utilização dos termos que descrevem seu contexto de execução e os recursos com os quais estas aplicações interagem.

Em suma, a criação da Ontologia de Recursos e da infraestrutura desenvolvida para o OCDRF atendem aos seguintes objetivos:

- fornecer à aplicação cliente acesso às informações de um recurso sem que esta precise utilizar exatamente o nome do tipo de recurso e o nome de seus atributos, usados durante o registro do mesmo, para localizá-lo; bastando para isto que seja definida uma correlação semântica entre os termos a serem usados e os termos registrados;

- fornecer um serviço de buscas por recursos que possuam determinadas capacidades, através da análise de sua composição, e não somente buscas por tipos específicos de recursos e seus super tipos;
- fornecer serviços capazes de responder à aplicação cliente utilizando os nomes de atributos e de capacidades conforme especificados na consulta, ao invés de retornar os nomes conforme registrado;
- e fornecer para a aplicação cliente um mecanismo para que conceitos de seu domínio, não definidos no núcleo deste projeto, sejam utilizados; através de pontos de extensão para a utilização de outras ontologias.

2.3.1 Utilização de Ontologias

Foi desenvolvida neste trabalho uma ontologia de aplicação, segundo a definição de (Guarino, 1997), denominada Ontologia de Recursos, para representar os recursos do contexto de execução das aplicações e os atributos de seus componentes. Esta ontologia é utilizada pelo OCDRF para a função de registro e armazenamento dos recursos na infraestrutura.

A Ontologia de Recursos, detalhada no Capítulo 3, contém a classificação de recursos (equipamentos eletrônicos e componentes de *software*) que podem ser encontrados em um ambiente ciente de contexto. A descrição de cada recurso contém:

(i) suas diversas denominações e equivalências, pelas quais poderá ser buscado,

(ii) sua composição em sensores e atuadores,

(iii) a classificação e hierarquia destes sensores e atuadores,

(iv) os atributos presentes em cada classe de sensores e atuadores e

(v) as diversas denominações e equivalências destes atributos.

Além disso, a Ontologia de Recursos representa também propriedades que estão diretamente ligadas ao recurso ou aos seus componentes, como por exemplo, os mecanismos de comunicação suportados pelos mesmos, i.e. suas APIs.

A Ontologia de Recursos definida para o OCDRF não se manterá estática durante a utilização da infraestrutura. Ela será “alimentada” pelo OCDRF durante a execução dos sistemas cientes de contexto. Conforme novos recursos forem registrados, informando seus

componentes e seus respectivos atributos, estes novos elementos serão validados e incluídos automaticamente pelo OCDRF na Ontologia de Recursos. Esta característica dispensa qualquer trabalho manual por parte do administrador da infraestrutura para o suporte a novos tipos de recursos, sendo este um diferencial da presente abordagem (ver Seção 6.2).

O OCDRF também disponibiliza pontos de extensão para que as aplicações clientes possam fazer referências e utilizar em suas buscas outras ontologias além da Ontologia de Recursos. Isto torna ainda mais ricas as possibilidades de utilização da infraestrutura e de buscas por parte da aplicação cliente, sendo este outro diferencial da presente abordagem (ver Seção 6.2). A utilização de Ontologias Cliente será descrita ao longo deste capítulo.

2.3.2 Integração de Recursos à Infraestrutura

Para poder se integrar à infraestrutura, um Agente de Recursos deve enviar um pedido de registro para o Serviço de Registro e Diretório. Este pedido deve incluir a descrição, utilizando a linguagem OWL, das características do recurso que ele representa.

A descrição de um recurso deve conter instâncias das classes definidas na Ontologia de Recursos, descritas no Capítulo 3, seguindo a especificação destas classes (relações obrigatórias, cardinalidades mínimas e máximas, etc). Durante a definição de um recurso também podem ser definidas novas classes e propriedades para melhor descrever o mesmo. Tais classes precisam então se relacionar com as classes já definidas na Ontologia de Recursos, sejam as classes definidas neste projeto, ou outras classes já registradas por outros ARs na Ontologia de Recursos. A partir da definição destas novas classes, as instâncias que representam o recurso podem então ser associadas a elas.

A modelagem e documentação das novas classes e propriedades a serem definidas na Ontologia de Recursos são de responsabilidade do administrador da infraestrutura. Sendo assim possível que outros reutilizem tais definições.

Na descrição de um recurso também pode ser efetuada a conexão com instâncias das Ontologias Cliente a serem usadas pelas aplicações clientes. Isto é, na definição de um recurso pode-se dizer, por exemplo, que o mesmo está “localizado” no “Quarto1”, sendo o termo “localizado” uma propriedade e “Quarto1” o identificador de uma instância, ambos definidos em uma Ontologia de Localização, um exemplo Ontologia Cliente, como denominado neste trabalho.

Em resumo, os elementos definidos durante o processo de registro dos ARs devem utilizar as classes básicas definidas pela Ontologia de Recursos ou podem estabelecer novas associações com as classes já registradas na Ontologia de Recursos. Podem ser estabelecidas correlações entre elementos recém chegados no sistema com os elementos já registrados.

Os ARs são representados como *Resources*, que irão possuir seus *ResourceComponents*, materializados em *Sensors* e *Actuators*. O Código 3 apresenta fragmentos da representação OWL utilizada para o registro de um recurso. O recurso apresentado contém um sensor e um atuador, e precisa ser implementado através de um Agente de Recurso, para seu funcionamento efetivo.

```
 1 <?xml version="1.0"?>
     ...
 2   <Resource rdf:ID="TV_LivingRoom">
        ...
 3      <hasComponent  rdf:resource="#TV_LivingRoom_Active_Engine" />
 4      <hasComponent  rdf:resource="#TV_LivingRoom_Write_Engine" />
 5      <loc:locatedIn rdf:resource="&loc;LivingRoom" />
 6   </Resource>
 7   <Sensor rdf:ID="TV_LivingRoom_Active_Engine">
        ...
 8      <hasCapacity   rdf:resource="#ActiveDevice" />
 9   </Sensor>
10   <InputCapacity rdf:ID="ActiveDevice">
11      <hasAttribute  rdf:resource="#powerOn" />
12   </InputCapacity>
13   <Actuator rdf:ID="TV_LivingRoom_Write_Engine">
14      <hasCapacity   rdf:resource="#TextOutput" />
15   </Actuator>
     ...
16 </rdf:RDF>
```

Código 3 - Descrição de um Recurso

As Linhas 2-6 do Código 3 apresentam a definição de um recurso que possui dois componentes. Estes componentes são representados nas Linhas 7-9 e 13-15. A Linha 5 apresenta a definição de uma conexão com uma instância de uma Ontologia Cliente. Os elementos da Ontologia de Recursos e os pontos de extensão para Ontologias Cliente serão detalhados no Capítulo 3.

O Código 4 apresenta fragmentos do registro de um recurso, contendo a definição de novas classes na Ontologia de Recursos.

```
 1 <?xml version="1.0"?>
     ...
 2 <owl:Class rdf:ID="OutputCapacity">
 3   <owl:oneOf rdf:parseType="Collection">
 4     <ExecutionCapacity rdf:about="#TextOutput" />
 5     <ExecutionCapacity rdf:about="#AudioOutput" />
 6     <ExecutionCapacity rdf:about="#VideoOutput" />
 7     <ExecutionCapacity rdf:about="#SMSOutput" />
 8   </owl:oneOf>
 9 </owl:Class>
10 <owl:Class rdf:ID="Output">
11   <rdfs:subClassOf rdf:resource="#Actuator" />
12   <rdfs:subClassOf>
13     <owl:Restriction>
14            <owl:onProperty rdf:resource="#hasCapacity" />
15            <owl:allValuesFrom rdf:resource="#OutputCapacity" />
16     </owl:Restriction>
17   </rdfs:subClassOf>
18 </owl:Class>
19 <Output rdf:ID="TV_LivingRoom_Write_Engine">
20   <hasCapacity  rdf:resource="#TextOutput" />
21 </Output>
     ...
22 </rdf:RDF>
```

Código 4 - Definição de novas classes durante a descrição de um Recurso

O Código 4 apresenta a definição das classes *OutputCapacity* (Linhas 2-9) e *Output* (Linhas 10-18). Estas classes determinam corelações com as classes *ExecutionCapacity* e *Actuator* definidas originalmente na Ontologia de Recursos (vide Figura 4, Seção 3.1). A definição da classe *OutputCapacity* determina completamente os indivíduos desta classe, que são instâncias de *ExecutionCapacity* e a definição da classe *Output* determina uma restrição sobre a propriedade *hasCapacity*. Apenas instâncias de *OutputCapacity* podem ser associadas a instâncias de *Output* através da propriedade *hasCapacity*.

As Linhas 19-21 apresentam a definição de uma instância que já utiliza a definição da nova classe *Output*. A definição desta instância pode então ser usada em substituição à definição utilizada nas Linhas 13-15 do Código 3, para a descrição do componente deste recurso.

A partir da definição destes novos termos, todos os registros de novos recursos poderão fazer uso dos mesmos e estes novos termos serão posteriormente utilizados normalmente pelo SRD durantes suas consultas aos recursos registrados, sem que seja necessário qualquer outro tipo de configuração. Cabe ao administrador da infraestrutura

documentar novas classes e propriedades criadas na Ontologia de Recursos para facilitar sua reutilização para a descrição de novos recursos.

Finalmente, uma vez que os recursos tenham sido registrados na infraestrutura, o Serviço de Descoberta fornecido pelo OCDRF pode ser acionado pela aplicação cliente para que tais recursos sejam localizados. O Código 5 apresenta um exemplo de consulta ao SD.

```
1 <DiscoveryQuery>
2     <CapacityConstraint ComponentType="Sensor" CapacityName="EquipamentoAtivo">
3         <Attribute Name="ligado" op="==" Value="true" />
4     </CapacityConstraint>
5 </DiscoveryQuery>
```

Código 5 - Exemplo de consulta com análise semântica ao SD

O Código 5 apresenta a busca por um recurso que possua como componente um sensor, com capacidade descrita como "*EquipamentoAtivo*" (Linha 2) e que possua um atributo denominado "ligado" que esteja com seu valor definido como verdadeiro (Linha 3). Caso tenha sido efetuada a ligação semântica entre os termos "*EquipamentoAtivo*" e "*ActiveDevice*" e entre os termos "ligado" e "*powerOn*", o Serviço de Descoberta será capaz de retornar o recurso registrado conforme descrito no Código 3, como resposta para a consulta efetuada no Código 5. Este tipo de resultado pode ser alcançado graças à análise semântica efetuada sobre as consultas recebidas pela infraestrutura e demonstra uma das vantagens da utilização das alterações propostas.

2.4 Serviços de Contexto com Ontologias

Durante a análise sobre as alterações que seriam implementadas pelo OCDRF sobre a infraestrutura original, decidiu-se por manter a arquitetura que divide as funções providas para as aplicações clientes pelos Serviços de Contexto, Descoberta e Registro e Diretório. Os motivos para a manutenção desta estrutura são os seguintes: (i) escalabilidade das funções de descoberta e contexto, (ii) possibilidade de extensão da função de diretório em federações deste serviço e (iii) manutenção do foco e simplicidade na implementação de cada função.

Considerando as funções de descoberta e contexto separadas em dois serviços, mantém-se a possibilidade de que vários Serviços de Descoberta e vários Serviços de

Contexto sejam inicializados em máquinas diferentes para atender a solicitações mais pesadas em termos de tempo de processamento ou em número de requisições.

No âmbito da aplicação cliente, a partir da necessidade de crescimento do número de serviços atendendo a esta, pode então ser construído um balanceador de cargas. Este seria responsável por obter junto ao SRD a listagem de Serviços de Descoberta e Contexto disponíveis no contexto de execução da aplicação e se encarregaria de lhes enviar as solicitações recebidas, de acordo com o algoritmo de balanceamento adotado.

O outro motivo para a manutenção da separação das funções providas pelo OCDRF em três serviços é o ponto de extensão descrito em (RODRIGUES, 2009a) para a utilização de federações de Serviços de Registro e Diretório. Cada SRD pode ser ligado a outros SRDs, de forma transparente para os clientes da infraestrutura, formando uma federação. Esta característica aumenta ainda mais o escopo das consultas que podem ser realizadas.

A partir da configuração de federações, o SRD adquire também a escalabilidade que alguns ambientes de execução podem exigir, sendo capaz de retornar os dados registrados e mantidos por outros SRDs da federação.

A arquitetura modular do CDRF facilitou a introdução das camadas necessárias para o tratamento das ontologias. Os componentes do OCDRF e as possíveis interações entre os mesmos são apresentados na Figura 3 e discutidos nas seções a seguir.

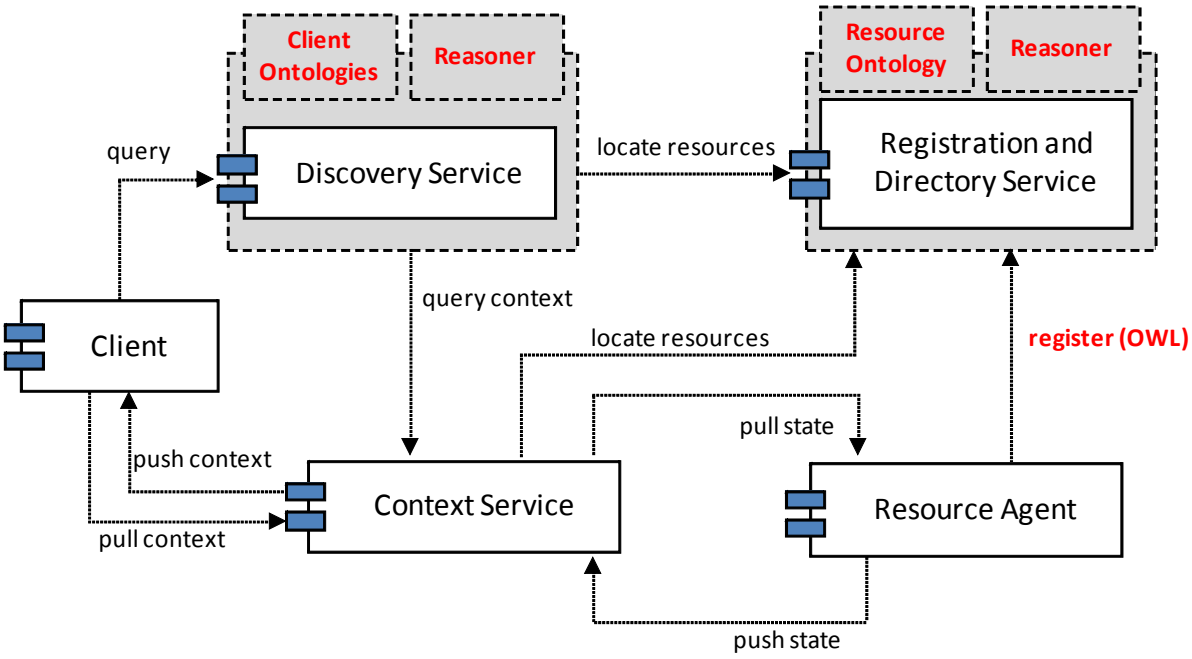

Figura 3 - Componentes do OCDRF

A seguir é apresentado o detalhamento das alterações propostas sobre cada componente da infraestrutura.

2.4.1 Serviço de Registro e Diretório

O SRD foi completamente remodelado para utilizar ontologias. A Ontologia de Recursos se tornou responsável pela função de diretório do SRD e todas as manipulações sobre ela ocorrem no SRD.

O SRD é o responsável pela análise de cada solicitação de registro e atualização automática da Ontologia de Recursos com as novas classes e instâncias recebidas. Esta função se dá sem a necessidade de nenhuma configuração prévia por parte do administrador do sistema.

O SRD permite que a utilização da Ontologia de Recursos seja enriquecida com regras de inferência. Assim, a Ontologia de Recursos pode ser associada a um conjunto de regras de inferência, que podem ser estabelecidas pelo administrador da infraestrutura. As relações extraídas com base nestas regras são a partir daí também analisadas antes do SRD responder a uma requisição. O Código 6 apresenta uma configuração que pode ser utilizada para a inicialização do SRD.

```
 1 <?xml version="1.0"?>
 2 <DirectoryService>
 3  <Description>Directory service</Description>
 4  <Ontology File="/conf/Resource.owl" Rules="/conf/ResourceRules.txt" />

 5  <Tecnologies>
 6    <Tecnology type="webservice" url=
          "http://localhost:1974/directory/DirectoryService?wsdl"/>
 7  </Tecnologies>

 8  <DiscoveryOntologies>
 9    <DiscoveryOntology Property="locatedIn"
          NS="http://localhost/ocdrf/conf/Location.owl#" />
10  </DiscoveryOntologies>

11 </DirectoryService>
```

Código 6 - Exemplo de Configuração do Serviço de Registro e Diretório

A Linha 4 do Código 6 apresenta a especificação do caminho para o arquivo que contém a Ontologia de Recursos e o caminho para o arquivo que contém as regras que devem

ser aplicadas a esta ontologia. Durante a sua inicialização o SRD efetua a carga desta ontologia e regras para memória.

As Linhas 8-10 do Código 6 apresentam um exemplo de configuração de Ontologias Cliente, detalhadas no Capítulo 3, que podem ser usadas no ambiente. Para o casamento de termos destas ontologias, o SRD precisa conhecer o *NameSpace* (NS) utilizado por estas. Isto é configurado como na Linha 9. Cada propriedade de Ontologias Cliente que tenha relação com as classes da Ontologia de Recursos e seja disponibilizada para consultas, deve ser configurada na inicialização do SRD. O exemplo do Código 6 apresenta na Linha 9, a configuração da propriedade "*locatedIn*".

O SRD suporta quantas Ontologias Cliente forem necessárias. Ele utiliza o NS especificado em sua configuração apenas para consultas à própria Ontologia de Recursos, onde as propriedades especificadas possuem seus relacionamentos com os recursos do contexto de execução das aplicações clientes. Não há portanto a necessidade de que seja feita a carga das Ontologias Cliente no SRD. A análise destas ficará a cargo do Serviço de Descoberta.

A configuração do SRD deve estar de acordo com a especificação apresentada no Código 7.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       attributeFormDefault="unqualified" elementFormDefault="qualified">
 3 <xsd:element name="DirectoryService">
 4 <xsd:complexType>
 5  <xsd:all>
 6    <xsd:element name="Description" minOccurs="0" />
 7    <xsd:element name="Ontology">
 8    <xsd:complexType>
 9       <xsd:attribute name="File" type="xsd:string" use="required" />
10       <xsd:attribute name="Rules" type="xsd:string" use="required" />
11    </xsd:complexType>
12    </xsd:element>

13   <xsd:element name="Technologies">
14   <xsd:complexType>
15   <xsd:sequence>
16         <xsd:element name="Technology" minOccurs="0" maxOccurs="unbounded">
17         <xsd:complexType>
18          <xsd:attribute name="type" type="xsd:string" use="required" />
19          <xsd:attribute name="url" type="xsd:string" use="required" />
20         </xsd:complexType>
21         </xsd:element>
22   </xsd:sequence>
23   </xsd:complexType>
```

```
24   </xsd:element>

25   <xsd:element name="DiscoveryOntologies">
26   <xsd:complexType>
27   <xsd:sequence>
28         <xsd:element name="DiscoveryOntology" minOccurs="0" maxOccurs="unbounded">
29         <xsd:complexType>
30          <xsd:attribute name="Property" type="xsd:string" use="required" />
31          <xsd:attribute name="NS" type="xsd:string" use="required" />
32         </xsd:complexType>
33         </xsd:element>
34   </xsd:sequence>
35   </xsd:complexType>
36   </xsd:element>
37  </xsd:all>
38 </xsd:complexType>
39 </xsd:element>
40 </xsd:schema>
```

Código 7 - Especificação da Configuração de um SRD

A Linha 6 do Código 7 apresenta uma descrição para leitura humana. As Linhas 7-12 especificam as configurações da Ontologia de Recursos. As Linhas 13-24 especificam as tecnologias suportadas pelo SRD e as Linhas 25-36 especificam as configurações das Ontologias Cliente usadas.

O Código 8 apresenta a especificação das consultas ao SRD.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     attributeFormDefault="unqualified" elementFormDefault="qualified">
 3 <xsd:element name="DirectoryQuery">
 4 <xsd:complexType>
 5    <xsd:all>
 6     <xsd:element name="Target">
 7     <xsd:complexType>
 8        <xsd:sequence>
 9            <xsd:element name="Attribute" minOccurs="0" maxOccurs="unbounded">
10              <xsd:complexType>
11               <xsd:attribute name="name" type="xsd:string" use="required" />
12              </xsd:complexType>
13            </xsd:element>
14        </xsd:sequence>
15        <xsd:attribute name="id" type="xsd:string" />
16        <xsd:attribute name="componentType" type="xsd:string" />
17        <xsd:attribute name="capacityName" type="xsd:string" use="required" />
18     </xsd:complexType>
19     </xsd:element>
20     <xsd:element name="includeFederation" type="xsd:boolean" minOccurs="0"
default="false"/>
21   </xsd:all>
```

```
22 </xsd:complexType>
23 </xsd:element>
24 </xsd:schema>
```

Código 8 - Especificação de uma consulta ao SRD

A consulta especificada no Código 8 poderá ser recebida pelo SRD através do Serviço de Contexto ou do Serviço de Descoberta. As Linhas 6-19 especificam o alvo da consulta. Apenas um alvo será enviado por vez ao SRD pelo SC e SD. O identificador "*id*" presente na Linha 15 é opcional. Ele só será preenchido caso o Serviço de Descoberta já possua uma listagem de possíveis resultados, com base em outras especificações recebidas em sua consulta ou quando o Serviço de Contexto estiver realizando a consulta.

Os atributos que se deseja buscar devem ser especificados conforme apresentado nas Linhas 9-13.

A Linha 16 apresenta o elemento "*componentType*" que pode ser do tipo "*Sensor*" ou "*Actuator*", a Linha 17 apresenta o elemento "*capacityName*" que representa a capacidade buscada e a Linha 20 apresenta o elemento opcional que solicita que a consulta seja propagada para todos os SRDs da federação.

O Código 9 apresenta a especificação das respostas retornadas pelo SRD. Os outros serviços da infraestrutura também utilizam a mesma especificação para as suas respostas.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       attributeFormDefault="unqualified" elementFormDefault="qualified">
 3 <xsd:element name="Response">
 4   <xsd:complexType>
 5   <xsd:sequence>

 6    <xsd:element name="ResourceInfo" minOccurs="0" maxOccurs="unbounded">
 7    <xsd:complexType>
 8      <xsd:sequence>
 9       <xsd:element name="InvokeMethod" minOccurs="0">
10       <xsd:complexType>
11         <xsd:attribute name="invokeString" type="xsd:string" use="required" />
12         <xsd:attribute name="operationName" type="xsd:string" use="required" />
13         <xsd:attribute name="returnType" type="xsd:string" use="required" />
14       </xsd:complexType>
15       </xsd:element>

16       <xsd:element name="Attribute" minOccurs="0" maxOccurs="unbounded">
17       <xsd:complexType>
18         <xsd:attribute name="name"  type="xsd:string" use="required" />
19         <xsd:attribute name="type"  type="xsd:string" use="required" />
```

```
20          <xsd:attribute name="unit"  type="xsd:string" use="required" />
21          <xsd:attribute name="value" type="xsd:string" />
22        </xsd:complexType>
23        </xsd:element>
24       </xsd:sequence>

25       <xsd:attribute name="id" type="xsd:string" use="required" />
26       <xsd:attribute name="componentType" type="xsd:string" use="required" />
27       <xsd:attribute name="capacityName" type="xsd:string" use="required" />

28     </xsd:complexType>
29     </xsd:element>
30    </xsd:sequence>
31    </xsd:complexType>
32 </xsd:element>
33 </xsd:schema>
```

Código 9 - Especificação das respostas do OCDRF

Os recursos localizados pelo SRD, SD e SC são expressados segundo as Linhas 6-29 do Código 9. Vários recursos podem ser retornados juntos pelos serviços providos pelo OCDRF (Linha 6). O atributo "*id*" (Linha 25) representa o identificador do recurso localizado e os atributos "*componentType*" (Linha 26) e "*capacityName*" (Linha 27) representam respectivamente o tipo de componente retornado, i.e. "*Sensor*" ou "*Actuator*" e o nome da capacidade que atende às especificações da consulta.

Para cada componente encontrado, são retornados os atributos que atendem às especificações da consulta (Linhas 16-23). Para cada atributo, são retornados "*name*" (Linha 18) que representa seu nome, "*type*" (Linha 19) que representa seu tipo, estático ou dinâmico, "*unit*" (Linha 20) que representa a unidade em que o valor do atributo é expresso e "*value*" (Linha 21) que representa o valor do atributo. Caso o atributo seja estático, o SRD já poderá retornar diretamente seu valor final.

O SRD é capaz de responder também a outro formato de consulta. Este formato, denominado consulta estendida neste trabalho, suporta as associações com Ontologias Cliente configuradas como no Código 6, isto é, suporta os pontos de extensão oferecidos pela infraestrutura.

O Código 10 apresenta a especificação deste tipo de consulta ao SRD.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      attributeFormDefault="unqualified" elementFormDefault="qualified">
3 <xsd:element name="EstendedDirectoryQuery">
```

```
 4  <xsd:complexType>
 5  <xsd:sequence>

 6   <xsd:element name="Target">
 7   <xsd:complexType>
 8       <xsd:attribute name="id" type="xsd:string" />
 9       <xsd:attribute name="element" type="xsd:string" use="required" />
10       <xsd:attribute name="property" type="xsd:string" use="required" />
11   </xsd:complexType>
12   </xsd:element>

13  </xsd:sequence>
14  </xsd:complexType>
15 </xsd:element>
16 </xsd:schema>
```

Código 10 - Especificação de uma consulta estendida ao SRD

As Linhas 6-12 especificam o alvo da consulta. Assim como na consulta base ao SRD, apenas um alvo poderá ser enviado por vez. O atributo "*id*" presente na Linha 8 representa o identificador de um recurso ou atributo de recurso, "*Resource*" ou "*Attribute*" respectivamente, conforme definido através do atributo "*element*" (Linha 9). O atributo "*id*" (Linha 8) é opcional, isto vem atender a consultas que queiram que sejam retornados todos os recursos que possuam determinada propriedade. Um exemplo de utilização de tal consulta é um *site survey* onde podem ser buscados todos os recursos que possuam a propriedade "*localizadoEm*".

Para finalizar, a Linha 10 apresenta o elemento "*property*" que representa a propriedade que estabelece a ligação entre a Ontologia de Recursos e a Ontologia Cliente. Para que esta propriedade seja localizada, ela deve estar configurada no SRD conforme especificado no Código 7.

O Código 11 apresenta a especificação das respostas do SRD às consultas estendidas.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       attributeFormDefault="unqualified" elementFormDefault="qualified">
 3 <xsd:element name="EstendedDirectoryResponse">
 4  <xsd:complexType>
 5  <xsd:sequence>

 6   <xsd:element name="TargetInfo" minOccurs="0" maxOccurs="unbounded">
 7   <xsd:complexType>
 8       <xsd:attribute name="id" type="xsd:string" use="required" />
 9       <xsd:attribute name="property" type="xsd:string" use="required" />
10       <xsd:attribute name="value" type="xsd:string" use="required" />
```

```
11    </xsd:complexType>
12    </xsd:element>
13   </xsd:sequence>
14   </xsd:complexType>
15 </xsd:element>
16 </xsd:schema>
```

Código 11 - Especificação da resposta do SRD a uma consulta estendida

Os recursos localizados pelo SRD são expressados segundo as Linhas 6-12. O atributo "*id*" (Linha 8) representa o identificador do recurso localizado. A Linha 9 apresenta a propriedade à qual a resposta se refere e a Linha 10 representa o valor encontrado para esta propriedade.

Com base nas respostas retornadas segundo o Código 9 e o Código 11, o Serviço de Descoberta e o Serviço de Contexto são capazes então de preparar suas respostas para a aplicação cliente.

2.4.2 Serviço de Descoberta

O Serviço de Descoberta é responsável por buscar informações sobre os recursos registrados na infraestrutura e a partir destas informações, analisar quais destes recursos estão de acordo com as condições especificadas nas consultas recebidas das aplicações clientes.

Através da utilização do SD proposto em (RODRIGUES, 2009a), uma aplicação cliente é capaz de encontrar um recurso através de sua identificação, o nome de seu tipo ou de seu super tipo. O novo SD realiza pesquisas baseado em análises semânticas e não apenas casamentos sintáticos, como era fornecido anteriormente. O Código 12 apresenta a busca por dispositivos registrados na infraestrutura, que estejam ligados, e que sejam capazes de exibir informações em formato de texto para o usuário.

```
1 <DiscoveryQuery>
2       <CapacityConstraint componentType="Actuator" capacityName="TextOutput">
3       </CapacityConstraint>
4       <CapacityConstraint componentType="Sensor" capacityName="ActiveDevice">
5             <Attribute name="powerOn" operator="==" value="true" />
6       </CapacityConstraint>
7 </DiscoveryQuery>
```

Código 12 - Exemplo de consulta ao Serviço de Descoberta do OCDRF

O Código 12 apresenta uma consulta de exemplo para o SD remodelado. Esta consulta espera como retorno a listagem dos recursos registrados no sistema que possuam componentes cujas capacidades sejam semanticamente iguais às duas solicitadas na consulta: "*TextOutput*" (Linhas 2-3) e "*ActiveDevice*" (Linhas 4-6), e cujo sensor com capacidade "*ActiveDevice*" possua um atributo semanticamente igual a "*powerOn*" (Linha 5) que esteja com o valor igual a "*true*", observados os parâmetros "*operator*" e "v*alue*". Na Ontologia de Recursos, as informações sobre os atributos dos recursos não estão diretamente ligados às capacidades, mas estão ligados aos atuadores e sensores, que estão ligados às capacidades, e é através desta correlação que a resposta é obtida (vide Figura 4).

As consultas enviadas ao SD devem seguir a especificação apresentada no Código 13.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       attributeFormDefault="unqualified" elementFormDefault="qualified">
 3 <xsd:element name="DiscoveryQuery">
 4 <xsd:complexType>
 5  <xsd:sequence>
 6   <xsd:element name="CapacityConstraint" minOccurs="0" maxOccurs="unbounded">
 7   <xsd:complexType>
 8     <xsd:sequence>
 9       <xsd:element name="Attribute" minOccurs="0" maxOccurs="unbounded">
10       <xsd:complexType>
11          <xsd:attribute name="name" type="xsd:string" use="required" />
12          <xsd:attribute name="operator" type="xsd:string" />
13          <xsd:attribute name="value" type="xsd:string" />
14          <xsd:attribute name="unit" type="xsd:string" />
15       </xsd:complexType>
16       </xsd:element>
17     </xsd:sequence>
18     <xsd:attribute name="componentType" type="xsd:string" use="required" />
19     <xsd:attribute name="capacityName" type="xsd:string" use="required" />
20   </xsd:complexType>
21   </xsd:element>
22   <xsd:element name="ClientConstraint" minOccurs="0" maxOccurs="unbounded">
23   <xsd:complexType>
24     <xsd:attribute name="element"   type="xsd:string" use="required" />
25     <xsd:attribute name="property"  type="xsd:string" />
26     <xsd:attribute name="name"      type="xsd:string" />
27     <xsd:attribute name="operation" type="xsd:string" use="required" />
28     <xsd:attribute name="object"    type="xsd:string" use="required" />
29   </xsd:complexType>
30   </xsd:element>
31  </xsd:sequence>
32  <xsd:attribute name="maxResults" type="xsd:integer" />
33 </xsd:complexType>
34 </xsd:element>
35 </xsd:schema>
```

Código 13 - Especificação de uma consulta ao Serviço de Descoberta

A consulta especificada no Código 13 poderá ser recebida pelo SD vinda das aplicações clientes. Ela suporta dois tipos de restrições: (i) uma relativa às características definidas na Ontologia de Recursos e (ii) outra relativa às características dos recursos relacionadas às ontologias definidas pela própria aplicação cliente, chamadas de Ontologias Cliente neste trabalho.

As restrições do tipo (i) são definidas como especificado nas Linhas 6-21 do Código 13. Os elementos das linhas 18 e 19 restringem respectivamente o tipo do componente e o nome da capacidade buscados. Os tipos de componentes são restritos a "*Sensor*" e "*Actuator*" e o nome da capacidade sofre análise semântica, i.e., instâncias registradas como "equivalentes" a buscada também são retornadas.

Os atributos que restringem a busca de cada recurso devem ser especificados conforme apresentado nas Linhas 9-16. O elemento opcional "*operator*" (Linha 12) representa a operação de comparação entre os atributos do recurso registrado e o valor representado pelo elemento "*value*" (Linha 13) cuja unidade pode ser expressa pelo elemento "*unit*" (Linha 14). Caso estes elementos opcionais não estejam preenchidos, o SD irá se certificar apenas de que o recurso retornado possua um atributo semanticamente igual ao elemento especificado conforme a Linha 11, "*name*". Caso nenhum atributo seja especificado na consulta, este não será um fator restritivo para a mesma.

As restrições do tipo (ii) são definidas como especificado nas Linhas 22-30. Elas representam o gancho disponibilizado pelo OCDRF para que ontologias definidas pelo próprio cliente sejam também analisadas pelo SD. A aplicação pode definir quantas restrições deste tipo forem necessárias e estas não são obrigatórias para a consulta (Linha 22). Todos os elementos que restringem a busca são do tipo texto (Linhas 24-28).

O atributo obrigatório "*element*" (Linha 24) pode se referir aos elementos "*Resource*" ou "*Attribute*" da Ontologia de Recursos. O atributo "*property*" (Linha 25) representa a propriedade que estabelece a conexão entre a Ontologia de Recursos e a Ontologia Cliente. Esta propriedade deve estar configurada na inicialização do SRD para que este serviço seja capaz de determinar a qual Ontologia Cliente esta propriedade se refere.

O atributo "*property*" (Linha 25) precisa ser definido apenas em pesquisas em que o atributo "*element*" (Linha 24) seja um "*Resource*". O atributo "*property*" irá representar o nome da propriedade adicionada à Ontologia de Recursos e associada à classe *Resource* durante a descrição de um recurso.

Já em pesquisas em que o atributo "*element*" (Linha 24) seja um "*Attribute*", o atributo "*property*" (Linha 25) não será utilizado. Nenhuma nova propriedade precisa ser adicionada à Ontologia de Recursos e associada à classe *Attribute* para este tipo de pesquisa. A propriedade a ser pesquisada na Ontologia de Recursos neste caso, será sempre a "*hasValue*", a ser apresentada no Capítulo 3. Para este tipo de pesquisa, o atributo "*name*" (Linha 26) deve ser definido. Ele representa o nome do atributo cujo valor deve ser utilizado para a comparação com a Ontologia Cliente.

O atributo obrigatório "*operation*" (Linha 27) representa a operação para a comparação na Ontologia Cliente, isto é, trata-se uma propriedade definida dentro da Ontologia Cliente que servirá para o batimento solicitado pela consulta. Esta propriedade deve estar configurada na inicialização do SD para que este serviço seja capaz de determinar a qual Ontologia Cliente esta propriedade se refere.

Para finalizar, o atributo obrigatório "*object*" (Linha 28) se refere ao objeto da comparação. Este deve ser um elemento definido na Ontologia Cliente.

Quando o SD recebe uma consulta, ele acessa o SRD para obter as informações registradas de acordo com cada restrição da consulta. De posse da resposta, o SD pode realizar o batimento entre os atributos solicitados e os atributos recebidos. A partir deste batimento, o SD é capaz então de obter a resposta final, que é formatada e enviada para o aplicativo cliente.

Uma aplicação em uma grande fábrica pode, por exemplo, precisar monitorar os termômetros que estejam medindo uma temperatura superior a 20ºC, que estejam localizados próximo à sua sala cofre. Para isto a aplicação pode efetuar uma consulta ao OCDRF como a listada no Código 14. Ao receber esta solicitação o SD irá obter junto ao SRD todas as instâncias de recursos registrados capazes de medir a temperatura do ambiente. O SD então acessará o SC e irá obter as medições mais recentes desses sensores, uma vez que a temperatura do ambiente seria, obviamente, um atributo dinâmico.

A resposta irá conter atributos, possivelmente em diferentes formatos, ou seja, os nomes dos atributos com a resposta de temperatura podem não ser sintaticamente iguais aos solicitados. Por isto, o SD precisa converter os termos das respostas para os termos esperados pela aplicação cliente. Além disto, o SD também poderia realizar a conversão das unidades dos atributos retornados pelo SC, para poder efetuar as comparações necessárias entre os valores nas unidades corretas, e poder retornar tais atributos conforme a unidade solicitada

pela aplicação cliente. Para finalizar, o SD então formata e retorna a resposta para a aplicação cliente.

```
1 <DiscoveryQuery>
2     <CapacityConstraint componentType="Sensor" capacityName="TemperatureMonitor">
3        <Attribute name="temperature" operator=">" value="20" unit="C" />
4     </CapacityConstraint>
5     <ClientConstraint element="Resource" property="locatedIn"
         operation="closeTo" object="Room21" />
6 </DiscoveryQuery>
```

Código 14 - Exemplo de consulta ao SD que utiliza Ontologias Cliente

O Código 14 na sua Linha 5 utiliza o gancho disponibilizado pelo OCDRF para a utilização de ontologias externas à infraestrutura. Os recursos retornados devem possuir associados a eles a propriedade "*locatedIn*" cujo valor deve estar associado ao elemento da Ontologia Cliente "*Room21*", que representa a sala cofre deste exemplo, através da propriedade "*closeTo*", que também deve estar definida na Ontologia Cliente.

O Código 15 apresenta um exemplo de resposta à consulta efetuada no Código 14.

```
1 <Response>
2    <ResourceInfo id="termometer056" componentType="Sensor"
       capacityName="TemperatureMonitor">
3       <Attribute name="temperature" type="Dynamic" unit="C" value="23" />
4    </ResourceInfo>
5    <ResourceInfo id="termometer059" componentType="Sensor"
       capacityName="TemperatureMonitor">
6       <Attribute name="temperature" type="Dynamic" unit="C" value="22" />
7    </ResourceInfo>
8 </Response>
```

Código 15 - Exemplo de resposta do Serviço de Descoberta

O Código 15 apresenta os dois recursos encontrados que estavam de acordo com a pesquisa efetuada no Código 14. As Linhas 2 e 5 do Código 15 apresentam a identificação dos recursos encontrados e as Linhas 3 e 6 apresentam os valores encontrados para o atributo *temperature* pesquisado.

Cada propriedade utilizada pela aplicação cliente em suas consultas, que faça referência a outras ontologias (que não sejam a Ontologia de Recursos) devem ser especificadas no arquivo de configuração do SD. Assim este pode identificar a qual ontologia cada propriedade se refere e tratar todas as Ontologias Cliente que estejam sendo utilizadas. O

Código 16 apresenta um exemplo de arquivo de configuração utilizado para a inicialização do SD.

```
 1 <?xml version="1.0" encoding="ISO-8859-1"?>
 2 <DiscoveryService>
 3 <Description>Discovery resources service</Description>
 4  <Ontology File="/conf/DS.owl" />
 5  <Tecnologies>
 6    <Tecnology type="webservice"
       url="http://localhost:1976/discovery/DiscoveryService?wsdl" />
 7  </Tecnologies>
 8  <Directories>
 9    <DirectoryService
       url="http://localhost:1974/directory/DirectoryService?wsdl"/>
10  </Directories>
11  <ContextService ID="ContextService01" />
12  <DiscoveryOntologies>
13      <DiscoveryOntology Property="locatedIn"
              NS="http://localhost/ocdrf/conf/Location.owl#"
              File="/conf/Location.owl"
              Rules="/conf/LocationRules.txt" />
14     <DiscoveryOntology Property="closeTo"
              NS="http://localhost/ocdrf/conf/Location.owl#"
              File="/conf/Location.owl"
              Rules="/conf/LocationRules.txt" />
15  </DiscoveryOntologies>
16 </DiscoveryService>
```

Código 16 - Exemplo de configuração do Serviço de Descoberta

A Linha 4 do Código 16 especifica o caminho do arquivo que contem a declaração do SD em OWL. Durante a sua inicialização o SD efetua seu registro junto ao SRD como faria qualquer AR que deseje se conectar à infraestrutura. As Linhas 5-7 definem o caminho onde deve ser inicializado o SD. As Linhas 8-10 apresentam a localização dos serviços de diretórios nos quais o SD deve se registrar. Para finalizar, a Linha 11 apresenta a identificação do Servico de Contexto que será utilizado. O mesmo será buscado a partir do SRD.

As Linhas 12-15 do Código 16 apresentam as propriedades não definidas na Ontologia de Recursos, que podem ser utilizadas nas consultas pela aplicação cliente, e a quais Ontologias Cliente estas propriedades se referem. Para o casamento de termos de outras ontologias, o SD precisa conhecer o *NameSpace* utilizado por estas. Este é definido pela propriedade "*NS*". O SD suporta quantas Ontologias Cliente e quantas propriedades realizando a conexão entre estas ontologias e a Ontologia de Recursos quantas forem necessárias.

Os recursos localizados pelo SD são expressados para a aplicação cliente segundo a mesma especificação usada para as respostas do SRD (Código 9, Seção 2.4.1).

2.4.3 Serviço de Contexto

O Serviço de Contexto é responsável por realizar a interface com os componentes dos Agentes de Recurso para obter seus estados atualizados e executar suas funcionalidades. O SC pode ser acessado pelo SD ou diretamente pelo aplicativo cliente. O SC aceita consultas para um recurso e componente específicos que tenham sido descobertos anteriormente e consultas compostas (que consideram referências a mais de um recurso e componente), caso em que os valores de todos os recursos deste tipo são retornados. O SC também aceita consultas assíncronas (também para um recurso e componente específicos).

O SC é capaz de identificar a origem das consultas que recebe, se estas vieram do SD, caberá ao SC apenas a extração dos dados dinâmicos de contexto mais recentes dos recursos envolvidos na consulta, através da comunicação com os respectivos ARs. Neste cenário, não caberá ao SC realizar nenhum tipo de conversão semântica ou checagem de consistência, dado que este trabalho já foi realizado pelo SD.

Já quando o SC recebe uma consulta direta de aplicações clientes, é sua responsabilidade a conversão semântica da consulta enviada e a conversão necessária para a preparação da resposta para a aplicação, utilizando os termos e formatos solicitados por esta. Neste cenário, o SC precisa se comunicar com o SRD para obter a definição original de cada recurso e realizar o casamento semântico entre os termos pesquisados e os termos registrados no sistema.

O Código 17 apresenta um exemplo de busca recebida de uma aplicação cliente.

```
1 <ContextQuery>
2   <synchronized>true</synchronized>
3   <Target id="Telcomed_WristClinic_AI067" capacityName="BloodPressure">
4      <Attribute Name="systolic"/>
5      <Attribute Name="diastolic"/>
6   </Target>
7 </ContextQuery>
```

Código 17 - Exemplo de consulta feita ao Serviço de Contexto do OCDRF

A consulta do Código 17 busca pelos valores atualizados de dois atributos (Linhas 4-5) do recurso de identificação "*Telcomed_WristClinic_AIO67*" (Linha 3) e que possui como uma de suas capacidades, uma denominada "*BloodPressure*" (Linha 3).

As consultas enviadas ao SC devem seguir a especificação apresentada no Código 18.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       attributeFormDefault="unqualified" elementFormDefault="qualified">
 3 <xsd:element name="ContextQuery">
 4 <xsd:complexType>
 5  <xsd:sequence>

 6   <xsd:element name="Target" minOccurs="0" maxOccurs="unbounded">
 7   <xsd:complexType>
 8     <xsd:sequence>

 9      <xsd:element name="InvokeMethod" minOccurs="0">
10      <xsd:complexType>
11        <xsd:attribute name="invokeString" type="xsd:string" use="required" />
12        <xsd:attribute name="operationName" type="xsd:string" use="required" />
13        <xsd:attribute name="returnType" type="xsd:string" use="required" />
14      </xsd:complexType>
15      </xsd:element>

16      <xsd:element name="Attribute" minOccurs="0" maxOccurs="unbounded">
17      <xsd:complexType>
18        <xsd:attribute name="name" type="xsd:string" use="required" />
19        <xsd:attribute name="unit" type="xsd:string" />
20      </xsd:complexType>
21      </xsd:element>

22     </xsd:sequence>
23     <xsd:attribute name="id" type="xsd:string" use="required" />
24     <xsd:attribute name="capacityName" type="xsd:string" use="required" />

25   </xsd:complexType>
26   </xsd:element>

27  </xsd:sequence>
28  <xsd:attribute name="requestFrom" type="xsd:string" />
29 </xsd:complexType>
30 </xsd:element>
31 </xsd:schema>
```

Código 18 - Especificação de uma consulta ao Serviço de Contexto

As consultas especificadas conforme o Código 18 poderão ser recebidas pelo SC vindas das aplicações clientes ou do Serviço de Descoberta. As Linhas 6-26 especificam o alvo da consulta. Esta consulta assume que o cliente ou SD já possui a identificação do

recurso e capacidade que lhe interessa. O identificador do recurso deve ser preenchido no atributo "*id*" (Linha 23) e a capacidade cujos valores de atributos se deseja buscar, deve ser informada como definido na Linha 24. Os atributos, buscados para cada recurso e capacidade, devem ser especificados conforme apresentado nas Linhas 16-21. Caso nenhum atributo seja especificado na consulta, todos os atributos configurados para o recurso serão retornados, sem que seja feito nenhum tratamento sobre seu nome ou unidade. Esta função pode ser útil para extração completa do *status* dos recursos do contexto de execução das aplicações.

A Linha 6 do Código 18 define que podem haver múltiplos alvos pesquisados em uma mesma consulta. Para cada alvo podem ser especificados os atributos que interessam à aplicação cliente ou ao SD e cada "*Attribute*" é composto do atributo obrigatório "*name*" (Linha 18), e do atributo opcional "*unit*" (Linha 19).

A aplicação cliente pode esperar receber como retorno do SC os valores dos atributos solicitados utilizando os mesmos termos enviados na consulta. Assim como o SD, o SC também é capaz de converter os termos das respostas para os termos esperados pela aplicação cliente. Além disto, o SC também poderia realizar a conversão das unidades dos atributos retornados pelos ARs, para poder retornar tais atributos conforme a unidade solicitada pela aplicação cliente.

A Linha 28 especifica a marcação que será enviada pelo SD para o SC. Esta marcação indica que os alvos que estão sendo requisitados ao SC já foram verificados pelo SD junto ao SRD e que as conversões semânticas necessárias para a montagem da resposta para o cliente serão de responsabilidade do SD. Caso esta marcação não seja recebida, o próprio SC fica encarregado de executar estas verificações e formatações.

Quando o SC recebe uma consulta vinda do SD, este já informa ao SD os detalhes obtidos junto ao SRD, necessários para a comunicação com os Agentes de Recurso. Estes são especificados conforme as Linhas 9-15 do Código 18. O método de invocação do AR, "*InvokeMethod*" (Linha 9) é composto pelos atributos obrigatórios do tipo texto "*invokeString*" (Linha 11), que representa a *string* de conexão com o AR, "*operationName*" (Linha 12), que representa o nome da operação que implementa o componente buscado e "*returnType*" (Linha 13), que representa o tipo de dado retornado pela operação.

Quando o SC recebe uma consulta vinda da aplicação cliente, ele acessa o SRD para obter as informações registradas. De posse da resposta, o SC pode agora fazer o batimento entre os atributos solicitados e os atributos recebidos. A partir deste batimento o SC obtém a

resposta final, que é formatada e enviada para o aplicativo cliente. As consultas realizadas para o SRD e respostas recebidas do mesmo, obedecem aos formatos estabelecidos como “consulta base” ao SRD, descritos no Código 8 e Código 9 respectivamente.

O Código 19 apresenta um exemplo de resposta à consulta efetuada no Código 17.

```
1 <Response>
2   <ResourceInfo id="Telcomed_WristClinic_AI067" componentType="Sensor"
       capacityName="BloodPressure">
3    <Attribute name="systolic"  type="Dynamic" unit="float" value="130" />
4    <Attribute name="diastolic" type="Dynamic" unit="float" value="80" />
5   </ResourceInfo>
6 </Response>
```

Código 19 - Resposta do Serviço de Contexto

O Código 19 apresenta a resposta do SC contendo os valores atualizados dos dois atributos solicitados na pesquisa efetuada.

As respostas do SC são expressadas para a aplicação cliente e para o SD, segundo a mesma especificação usada para as respostas do SRD e do SD (Código 9, Seção 2.4.1).

A unicidade dos formatos de respostas do SD e SC facilita o tratamento das respostas retornadas pelos dois serviços da infraestrutura com os quais a aplicação cliente se comunica para recuperar informações sobre os recursos do seu contexto de execução.

2.4.4 Agentes de Recurso

Os Agentes de Recurso (ARs) possuem a função de disponibilizar as operações descritas pelos componentes dos recursos conectados à infraestrutura, isto é, eles fornecem a implementação das capacidades descritas por um recurso.

Um AR representa a camada de *software* que implementa um *Resource*, capaz de se registrar na infraestrutura e de fornecer um meio de acesso aos seus componentes. Ele esconde os detalhes de implementação da comunicação com os seus sensores e atuadores.

Cada componente (sensor ou atuador) de um recurso deve possuir uma implementação fornecida por uma operação do AR. No processo de registro, as informações sobre a forma de acesso e os parâmetros de entrada e saída de tais operações também precisam ser fornecidas. Estas informações serão usadas posteriormente pelo Serviço de Contexto para interagir com

cada componente do recurso, a fim de extrair dados de contexto ou executar alguma das funções providas pelo recurso.

As informações de registro dos ARs no OCDRF não demandam formatação seguindo um XSD específico. Também não é necessário que os ARs sejam definidos apenas conforme tipos com estruturas previamente cadastradas pelo administrador do sistema como ocorria no CDRF. Não há mais a necessidade de registro de tipos e a única restrição para a formatação das informações de registro de um AR é que as mesmas estejam descritas em OWL 1.0, formato suportado pelo *framework* Jena2, que foi utilizado na implementação de referência deste projeto.

Durante o processo de registro, ao invés de uma *string* XML sem relevância semântica, os ARs remodelados enviam para o SRD uma descrição em OWL que representa suas características, seus componentes e suas funcionalidades de acordo com a Ontologia de Recursos. Durante sua inicialização, cada AR transfere sua descrição OWL, como a apresentada no Código 3, serializada para o SRD.

Para que a aplicação cliente seja capaz de localizar os recursos disponíveis no seu contexto de execução através do OCDRF, as descrições destes recursos precisam estar de acordo com a Ontologia de Recursos. Novas classes e propriedades que precisem ser definidas e as instâncias criadas para a descrição dos ARs devem obedecer à estrutura disponibilizada pela Ontologia de Recursos. Isto é, caso seja necessária a definição de novas classes e ou de novas propriedades, estas devem ser definidas como subclasses ou estabelecer novas relações com outras classes já incluídas na ontologia. Novas propriedades devem ser definidas como equivalentes ou inversas às outras já criadas, por exemplo. Além disto, as instâncias declaradas para representar os recursos devem pertencer às classes da Ontologia de Recursos ou às classes relacionadas a estas. Caso a descrição registrada por um AR não seja compatível com a Ontologia de Recursos, os componentes do recurso que ele implementa serão ignorados durante as consultas realizadas pelo SD ao SRD.

3 ONTOLOGIA DE RECURSOS

Este capítulo detalha como as informações dos recursos do contexto das aplicações clientes são representadas através de elementos da Ontologia de Recursos. Esta é a ontologia principal para o funcionamento do OCDRF e possui a representação de todos os recursos disponibilizados no contexto das aplicações clientes, além da representação dos próprios elementos da infraestrutura. São apresentados também os pontos de extensão disponibilizados pela infraestrutura para que as aplicações clientes possam se utilizar de outras ontologias que descrevam seu domínio de aplicação. No final deste capítulo é descrito como a Ontologia de Recursos pode ser estendida através da definição de regras de inferência.

3.1 Elementos da Ontologia

A Ontologia de Recursos representada na Figura 4 foi desenvolvida para dar suporte ao OCDRF. Através dela são representados os recursos disponíveis no contexto de execução das aplicações cientes de contexto.

Foram aplicadas tons de cinza e estilos de linhas diferentes em alguns elementos do diagrama apresentado na Figura 4 para facilitar sua visualização e leitura. Os elementos em cinza escuro representam as classes centrais ou mais importantes da ontologia. Os elementos em cinza claro representam classes secundárias na interpretação do diagrama.

O elemento em branco, tracejado, associado à classe *Attribute* através da propriedade *hasValue* representa a classe "*Thing*" especificada em OWL (W3C2, 2004). Ela é a raiz de todas as classes e literais. Todo indivíduo em OWL é um membro da classe *owl:Thing*. Então, por polimorfismo, onde *Thing* foi especificado no diagrama, qualquer elemento que seja especificado em uma ontologia, pode ser utilizado. Os elementos em branco e linhas contínuas representam literais que podem ser definidos utilizando OWL, isto é, *integer*, *nonNegativeInteger*, *string*, *boolean*, etc.

Para finalizar, os elementos destacados em cinza claro, tracejados, associados às classes *Resource* e *Attribute* representam os pontos de extensão (ganchos) fornecidos pela Ontologia de Recursos e pelo OCDRF para que as aplicações clientes expandam esta ontologia através da utilização de outras ontologias.

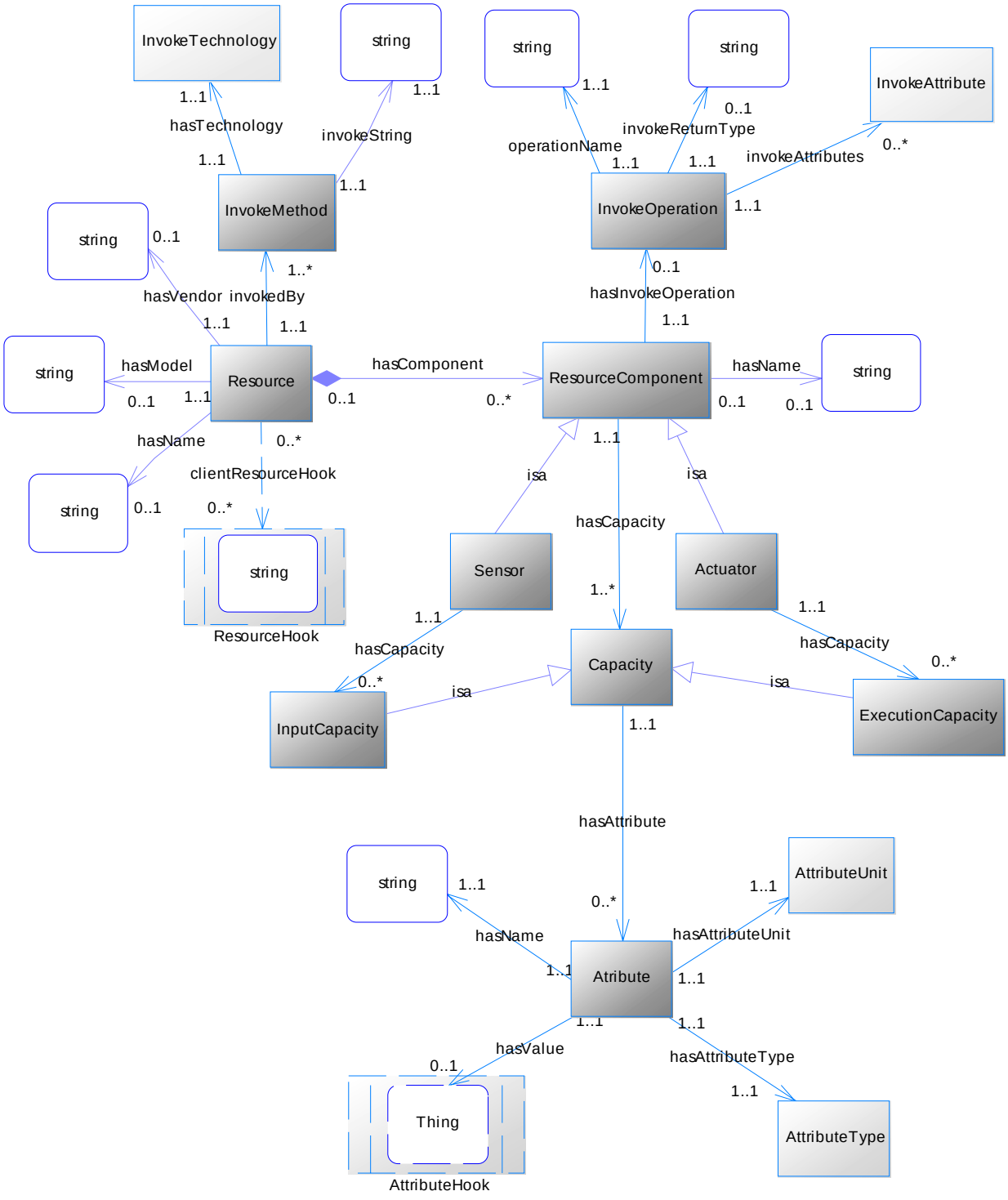

Figura 4 - Representação da Ontologia de Recursos

A seguir serão descritas as classes definidas na Ontologia de Recursos e representadas na Figura 4.

A classe *Resource*, representa um dispositivo eletrônico ou um componente de *software* com o qual a aplicação ciente de contexto irá interagir. Um *Resource* pode possuir

um ou mais componentes, "*ResourceComponents*", definidos através da propriedade *hasComponent*. Um *Resource* possui como informações opcionais um nome, modelo e fabricante. Além disto, um *Resource* possui associado a ele um gancho representado por "*ResourceHook*", para que a aplicação cliente possa estender a Ontologia de Recursos. Este gancho será detalhado na Seção 3.2.

A classe *ResourceComponent* possui como especializações as classes *Sensor* e *Actuator*. O termo *Sensor* representa um componente responsável por recuperar informações de estado do contexto de execução das aplicações (entradas para o sistema) enquanto o termo *Actuator* representa um componente capaz de causar alterações nas informações de estado do contexto (saídas do sistema).

Um *ResourceComponent* pode possuir um nome para leitura humana e deve estar associado a uma ou mais capacidades. As capacidades são representadas pela classe *Capacity* e são associadas aos componentes dos recursos através da propriedade *hasCapacity*.

As classes *Sensor* e *Actuator* possuem propriedades especializadas da propriedade *hasCapacity* que as associa às especializações da classe *Capacity*, respectivamente *InputCapacity* e *ExecutionCapacity*. *InputCapacity* representa a capacidade de monitoramento que um componente pode possuir e *ExecutionCapacity* representa a capacidade que um componente possui de executar outras operações, que não sejam atividades de extração de informações de contexto.

Os indivíduos das classes *InputCapacity* e *ExecutionCapacity* possuirão os detalhes que permitirão que buscas precisas sejam feitas pelo Serviço de Descoberta e que seja possível o posterior acesso a estas capacidades, seja para a extração de informações de contexto ou execução de algum outro tipo de função. As funções executadas por um atuador podem alterar o contexto ou não, como por exemplo, uma operação pode apenas apresentar uma informação para o usuário, não alterando o contexto de execução da aplicação.

O Código 20 apresenta a definição da classe *InputCapacity* herdando as características e relacionamentos da classe *Capacity* (Linha 2). A definição da classe *ExecutionCapacity* é feita de forma análoga.

```
1     <owl:Class rdf:ID="InputCapacity">
2           <rdfs:subClassOf rdf:resource="#Capacity" />
3     </owl:Class>
```

Código 20 - Exemplo de definição da classe *InputCapacity*

O Código 21 apresenta a definição da classe *Sensor* e a restrição que esta classe possui em relação à propriedade *hasCapacity* (Linhas 3-6) que determina que todos os valores definidos nesta propriedade devem pertencer à classe *InputCapacity* (Linha 5). A definição da classe *Actuator* é feita de forma análoga, porém utilizando como restrição a classe *ExecutionCapacity*.

```
1     <owl:Class rdf:ID="Sensor">
2         <rdfs:subClassOf>
3            <owl:Restriction>
4                  <owl:onProperty rdf:resource="#hasCapacity" />
5                  <owl:allValuesFrom rdf:resource="#InputCapacity" />
6            </owl:Restriction>
7         </rdfs:subClassOf>
8     </owl:Class>
```

Código 21 - Exemplo de definição da classe *Sensor*

Vale observar no Código 21 que a classe *Sensor* não é definida explicitamente como subclasse de *ResourceComponent*. Porém possui associada a ela a propriedade *hasCapacity* que foi definida na ontologia como propriedade que apenas um *ResourceComponent* possui, isto é, como uma propriedade cujo domínio é representado por *ResourceComponent*. Logo, dado que a classe Sensor, definida no Código 21, possui uma restrição relativa à propriedade *hasCapacity* (Linha 4), por inferência, esta classe é uma subclasse de *ResourceComponent*. Então, através desta herança "por inferência", a classe *Sensor* passa a possuir todas as propriedades definidas em *ResourceComponent*. Esta é uma das flexibilidades fornecidas pela linguagem OWL para a descrição das classes e propriedades da ontologia.

Uma instância da classe *Capacity* pode estar relacionada a diversos atributos através da propriedade *hasAttribute*. Estes são os dados do contexto de execução das aplicações a serem monitorados e utilizados pelas aplicações cientes de contexto. Um *Attribute* possui como informações obrigatórias um nome, um tipo (representado pela classe "*AttributeType*") e a unidade de medida na qual o valor do atributo será representado, representada pela classe "*AttributeUnit*".

Um *Attribute* também possui como informação opcional um valor, que pode assumir qualquer tipo, uma vez que foi declarado na ontologia como "*Thing*". A informação do valor do atributo deve ser especificada de acordo com a unidade de medida utilizada no atributo. Além disto, um *Attribute* possui associado a ele um gancho representado por "*AttributeHook*",

para que a aplicação cliente possa estender a Ontologia de Recursos. Este gancho será detalhado na Seção 3.2.

Objetos da classe *Attribute* devem estar associados a um *AttributeType* através da propriedade *hasAttributeType.* A classe *AttributeType* representa os tipos de atributos que a aplicação pode tratar. Estes podem ser apenas de tipo dinâmico ou estático conforme apresentado no Código 22.

```
1     <owl:Class rdf:ID="AttributeType">
2     ...
3         <owl:oneOf rdf:parseType="Collection">
4            <owl:Thing rdf:about="#Static" />
5            <owl:Thing rdf:about="#Dynamic" />
6         </owl:oneOf>
7     </owl:Class>
```

Código 22 - Definição da Classe *AttributeType*

O Código 22 apresenta a criação da classe *AttributeType* através da definição de uma coleção. A linguagem OWL permite que através desta definição sejam especificados completamente os indivíduos de uma classe, assim sendo, nenhum outro indivíduo pode ser declarado como membro da mesma.

As formas de invocação suportadas pela implementação de um *Resource* são designadas pela propriedade *invokedBy* que associa o recurso a uma ou mais instâncias da classe *InvokeMethod*. Esta classe possui um atributo denominado *invokeString* que conterá a *string* de invocação (uma URL caso a implementação seja feita utilizando *web services*, por exemplo). Além disto, a classe *InvokeMethod* também possui associada a ela a classe denominada *InvokeTechnology*, que representa a tecnologia de comunicação que pode ser utilizada para acessar o componente através deste *InvokeMethod*. O Código 23 apresenta um exemplo de definição de um indivíduo da classe *InvokeTechnology*.

```
1     <InvokeTechnology rdf:ID="webservice">
2           <hasName rdf:datatype="&xsd;string">webservice</hasName>
3     </InvokeTechnology>
```

Código 23 - Exemplo de definição de um indivíduo da classe *InvokeTechnology*

Cada componente de um recurso (sensor ou atuador) deve possuir associado a ele uma implementação, fornecida por uma operação do seu Agente de Recurso. Esta implementação é

representada pela classe *InvokeOperation* associada à classe *ResourceComponent*. Esta classe define o nome da operação, "*operationName*", o tipo de dado retornado por ela, "*invokeReturnType*", e os parâmetros de entrada para a mesma, "*invokeAttributes*", opcionais. Esta modelagem permite que um Agente de Recursos possua mais de uma operação de monitoramento, representando seus diversos sensores e mais de uma operação que altere ou não seu contexto, representando seus diversos atuadores.

A classe *Sensor* possui uma especialização da propriedade "*hasInvokeOperation*" que representa as operações de monitoramento do contexto da aplicação. Estas possuem necessariamente como parâmetro de saída uma variável do tipo texto e não possuem parâmetros de entrada. Esta especialização é apresentada na Figura 5.

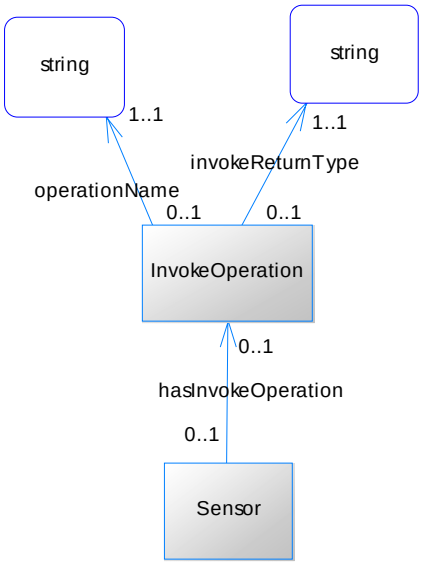

Figura 5 - Especialização da operação *hasInvokeOperation*

Além das relações representadas no diagrama da Figura 4, são definidas na Ontologia de Recursos as relações inversas *usesCapacity* e *isComponentOf*, respectivamente inversas às propriedades *hasCapacity* e *hasComponent*.

3.2 Pontos de Extensão

A Ontologia de Recursos desenvolvida para dar suporte ao OCDRF provê dois pontos de extensão para as aplicações clientes, representados pelos elementos em cinza pontilhados na Figura 4. Estes ganchos permitem que as aplicações clientes expandam a Ontologia de

Recursos através da utilização de outras ontologias. Estas ontologias são denominadas Ontologias Cliente neste trabalho.

O elemento *ResourceHook* conectado à classe *Resource* e o elemento *AttributeHook* conectado à classe *Attribute* através das propriedades *clientResourceHook* e *hasValue* respectivamente, não representam classes criadas e disponibilizadas inicialmente na Ontologia de Recursos. Estes elementos representam classes de ontologias criadas para as aplicações clientes para que estas tenham maior flexibilidade e possam utilizar termos do seu próprio domínio nas suas buscas aos serviços da infraestrutura. Assim, uma aplicação cliente é capaz de acrescentar mais semântica aos recursos descritos na ontologia principal.

O valor de um atributo de um recurso pode representar por exemplo um *status* relevante para a aplicação cliente. Através da criação de uma outra ontologia é possível dar sentido aos possíveis *status* que os recursos podem assumir e realizar consultas utilizando as relações definidas nesta ontologia.

Em uma aplicação responsável pela monitoração de servidores por exemplo, cada servidor pode ser representado por um *Resource* e possuir um Agente de Recursos executando para buscar informações atualizadas. O Código 24 apresenta fragmentos da especificação deste tipo de recurso.

```
 1  ...

 2  <Resource rdf:ID="Server003">
      ...
 3    <hasComponent       rdf:resource="#Disk_Monitor_Engine" />
 4  </Resource>

 5  <Sensor rdf:ID="Disk_Monitor_Engine">
      ...
 6    <hasCapacity        rdf:resource="#Disk_Monitor" />
 7  </Sensor>

 8  <InputCapacity rdf:ID="Disk_Monitor">
 9    <hasAttribute       rdf:resource="#status" />
10  </InputCapacity>

11  <Attribute rdf:ID="status">
12     <hasName           rdf:datatype="&xsd;string">status</hasName>
13     <hasAttributeType  rdf:resource="#Dynamic" />
14     <hasAttributeUnit  rdf:resource="#String" />
15  </Attribute>

16 </rdf:RDF>
```

Código 24 - Exemplo de definição de um recurso

Cada *status* retornado pelos recursos possui uma criticidade maior que outro para a aplicação cliente, assim como na especificação utilizada para o log4j (APACHE, 2011). Para representar a criticidade de cada *status*, uma ontologia pode ser definida como a apresentada no Código 25.

```
 1 <?xml version="1.0"?>
 2 <!DOCTYPE owl [
 3     <!ENTITY st  "http://localhost/ocdrf/conf/Status.owl#" >
 4     <!ENTITY owl  "http://www.w3.org/2002/07/owl#" >
 5     <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#" >
 6   ]>

 7 <rdf:RDF
 8  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 9  xmlns:rdfs= "http://www.w3.org/2000/01/rdf-schema#"
10  xmlns:owl = "&owl;"
11  xmlns:xsd = "&xsd;"
12  xml:base  = "&st;"
13  xmlns     = "&st;">

14 <owl:Ontology rdf:about="">
15     <owl:versionInfo>1.0</owl:versionInfo>
16     <rdfs:label>Status Ontology</rdfs:label>
17 </owl:Ontology>

18 <owl:Class rdf:ID="Status" />

19 <owl:ObjectProperty rdf:ID="betterThan">
20     <rdf:type rdf:resource="&owl;TransitiveProperty" />
21     <rdfs:domain rdf:resource="#Status" />
22     <rdfs:range rdf:resource="#Status" />
23 </owl:ObjectProperty>

24 <owl:ObjectProperty rdf:ID="worseThan">
25     <owl:inverseOf rdf:resource="#betterThan"/>
26 </owl:ObjectProperty>

27 <Status rdf:ID="FATAL" />

28 <Status rdf:ID="ERROR">
29     <betterThan rdf:resource="#FATAL" />
30 </Status>

31 <Status rdf:ID="WARN">
32     <betterThan rdf:resource="#ERROR" />
33 </Status>

34 <Status rdf:ID="INFO">
35     <betterThan rdf:resource="#WARN" />
36 </Status>

37 <Status rdf:ID="DEBUG">
38     <betterThan rdf:resource="#INFO" />
```

```
39 </Status>

40 </rdf:RDF>
```

Código 25 - Exemplo de Ontologia Cliente

Na ontologia apresentada no Código 25 são definidas uma classe denominada "*Status*" (Linha 18) e duas propriedades, denominadas "*betterThan*" e "*worseThan*" (Linhas 19-23 e 24-26), além das instâncias desta classe e da correlação entre elas.

A partir da definição e configuração desta ontologia no OCDRF, uma aplicação cliente seria capaz de realizar consultas ao SD buscando pelos recursos que possuam um atributo de nome *status*, cujo valor seja pior que ("*worseThan*") "*WARN*". Com base na propriedade inversa "*worseThan*" e na transitividade da propriedade "*betterThan*", será possível ao OCDRF retornar os recursos cujo atributo *status* esteja igual a *ERROR* ou *FATAL*.

Outro exemplo da utilização de Ontologias Cliente é apresentado na Seção 5.1. Neste exemplo os cômodos de uma residência são organizados em uma ontologia e associados aos recursos disponíveis no ambiente para serem utilizados por uma aplicação de telemedicina.

Para que o gancho fornecido pelo OCDRF possa ser utilizado, o administrador da aplicação cliente precisa efetuar os seguintes passos:

(i) criação das ontologias de interesse para o campo de atuação da aplicação;

(ii) configuração junto ao SRD de quais propriedades das Ontologias Cliente serão incluídas na ontologia principal, fazendo associação entre as mesmas e os seus *NameSpaces*, como apresentado no Código 26;

```
...
1  <DiscoveryOntologies>
2    <DiscoveryOntology Property="locatedIn"
          NS="http://localhost/ocdrf/conf/Location.owl#" />
3  </DiscoveryOntologies>
...
```

Código 26 - Configuração de propriedades de uma Ontologia Cliente no SRD

(iii) configuração junto ao SD de quais propriedades serão utilizadas para as consultas referenciando cada uma das Ontologias Cliente criadas, como apresentado no Código 27;

```
...
1  <DiscoveryOntologies>
2      <DiscoveryOntology Property="locatedIn"
              NS="http://localhost/ocdrf/conf/Location.owl#"
              File="/conf/Location.owl"
              Rules="/conf/LocationRules.txt" />
3      <DiscoveryOntology Property="closeTo"
              NS="http://localhost/ocdrf/conf/Location.owl#"
              File="/conf/Location.owl"
              Rules="/conf/LocationRules.txt" />
4  </DiscoveryOntologies>
...
```

Código 27 - Configuração de propriedades de uma Ontologia Cliente no SD

(iv) e inclusão de associações entre as classes das Ontologias Cliente com as classes *Resource* ou *Attribute* ou ainda com a super classe OWL *Thing*, na descrição dos recursos que venham a se registrar no SRD.

Assim que estes passos de configuração estejam concluídos, a aplicação cliente passa a ser capaz de efetuar consultas ao SD utilizando os termos de suas ontologias, como apresentado no Código 14.

O OCDRF suporta, através dos pontos de extensão disponibilizados, a utilização de quantas Ontologias Cliente as aplicações necessitem, sejam classes associadas aos Recursos ou aos Atributos destes recursos. Um exemplo detalhado contendo a criação e utilização de Ontologias Clientes é apresentado no Capítulo 5 deste trabalho.

3.3 Regras de Inferência

Além da Ontologia de Recursos foram definidas neste trabalho regras de inferência para facilitar as buscas providas pelo OCDRF. Estas regras precisam ser configuradas no SRD conforme apresentado no Código 28.

```
1  ...
2  <Ontology File="/conf/Resource.owl" Rules="/conf/ResourceRules.txt" />
3  ...
```

Código 28 - Configuração do arquivo contendo as Regras de Inferência

No Código 28 é definido o caminho para o arquivo onde estão as regras a serem aplicadas sobre a Ontologia de Recursos. Tais regras devem ser especificadas no formato esperado pelo Jena, como no exemplo apresentado no Código 29.

```
1  @prefix p: <http://localhost/ocdrf/conf/Resource.owl#>

2 [rule1: (?res p:hasComponent ?com) (?com p:hasCapacity ?cap) -> (?res p:hasCapacity
?cap)]
3 [rule2: (?A p:hasCapacity ?cap) (?cap p:hasAttribute ?at) -> (?A p:hasAttribute ?at)]
```

Código 29 - Configuração de regras para inferências sobre a Ontologia de Recursos

A primeira linha do Código 29 define o *NameSpace* a ser utilizado nas regras e o atribui à variável “p”. A Linha 2 define que se uma instância qualquer “*res*” da ontologia está associada a uma segunda instância “*com*” através da propriedade *hasComponent* e esta está associada a uma terceira instância “*cap*” através da propriedade *hasCapacity*, então, a primeira instância da relação, “*res*”, também está associada à terceira instância, “*cap*”, através da propriedade *hasCapacity*. Esta regra implica uma correlação direta entre os recursos e as capacidades de seus componentes, disponibilizando para as consultas do OCDRF uma relação que não estava definida originalmente na Ontologia de Recursos.

A regra seguinte, definida na Linha 3, se complementa à primeira regra definida, e a partir destas duas regras é possível então inferir uma associação direta entre um recurso e seus atributos. Relação esta muito utilizada pelas consultas realizadas pelo OCDRF.

4 IMPLEMENTAÇÃO DE REFERÊNCIA

Este capítulo apresenta a implementação de referência dos serviços propostos para o OCDRF. O *Ontology Contextual Discovery of Resources Framework* (OCDRF) foi criado com o objetivo de facilitar o desenvolvimento de novas aplicações cientes de contexto provendo a infraestrutura básica para sua execução, com suporte a consultas semânticas.

A linguagem OWL foi utilizada para definir a Ontologia de Recursos e as Ontologias Clientes e o *framework Jena* (JENA, 2011) foi utilizado para auxiliar na manipulação destas ontologias. A linguagem Java SE 6 foi utilizada para implementar as alterações nos serviços da infraestrutura.

O trabalho desenvolvido em (RODRIGUES, 2009a), que utiliza a XML Web Services API, JAX-WS 2.0 (JAX-WS, 2011) e o servidor HTTP embutido no conjunto de ferramentas disponibilizado pela plataforma Java SE 6, onde os serviços e os Agentes de Recursos são executados como Serviços *Web* independentes, foi utilizado como base para o desenvolvimento da nova infraestrutura.

4.1 Jena

Foi utilizada a linguagem OWL para o desenvolvimento das ontologias integradas ao OCDRF. A *Web Ontology Language*, ou OWL, é uma recomendação do W3C, que fornece um mecanismo para descrever formalmente um campo de conhecimento. Além de definir classes hierárquicas às quais os recursos podem pertencer, o OWL permite que as características das propriedades dos recursos possam ser expressas (MCCARTHY, 2004).

Para processar as ontologias descritas em OWL, o *framework* Jena (JENA, 2011) foi utilizado. O Jena - *Jena Semantic Web Framework* - é uma API Java para a manipulação de ontologias. Ele foi criado pela Hewlett-Packard (HP) e tornou-se *open source* (HP, 2009).

Utilizando o Jena, uma ontologia é tratada como um tipo especial de modelo RDF (W3C6, 2004), através da interface *OntModel*. Essa interface permite que a ontologia seja manipulada programaticamente, com métodos convenientes para criar classes, restrições das propriedades, e assim por diante (MCCARTHY, 2004).

Em Jena, o sujeito de uma declaração é sempre um *Resource*, o predicado é representado por um objeto da classe *Property* e o objeto pode ser um outro *Resource* ou um

valor literal. Literais são representados em Jena pelo tipo *Literal*. Todos estes tipos compartilham uma interface comum, *RDFNode*.

As consultas a um modelo Jena via programação são basicamente realizadas por meio de métodos *list*() das interfaces *Model* e *Resource*. Esses métodos podem ser utilizados para obter sujeitos, objetos e declarações correspondentes a determinadas condições. Eles também retornam especializações de *java.util.Iterator* (ORACLE2, 2011), que possuem métodos extra para retornar tipos específicos de objetos.

As capacidades de raciocínio do Jena podem ser usadas para inferir conhecimento a partir de uma ontologia. O Jena é capaz de derivar declarações adicionais que o modelo não expressa diretamente (MCCARTHY, 2004). Na distribuição do Jena, são incluídas várias máquinas de inferência pré-definidas (JENA, 2011) para trabalhar com diferentes tipos de ontologias.

4.2 Implementação do Serviço de Registro e Diretório

O serviço da infraestrutura que mais sofreu alterações em comparação ao proposto em (Rodrigues, 2009a) foi o Serviço de Registro e Diretório (SRD). Sua implementação inicial era baseada em listas de objetos em memória e foi totalmente remodelada para usar a Ontologia de Recursos como fonte de informações (diretório) dos recursos disponíveis.

Na solução adotada foi abolida a leitura de tipos de recursos pré-cadastrados pelo administrador da infraestrutura, anteriormente realizada na iniciação do SRD. A definição de tipos estáticos foi substituída pela definição de classes dentro da Ontologia de Recursos. Classes estas que são alimentadas ao longo da execução das aplicações cientes de contexto, através do registro de cada recurso na infraestrutura.

A Figura 6 apresenta a nova estrutura de classes do SRD.

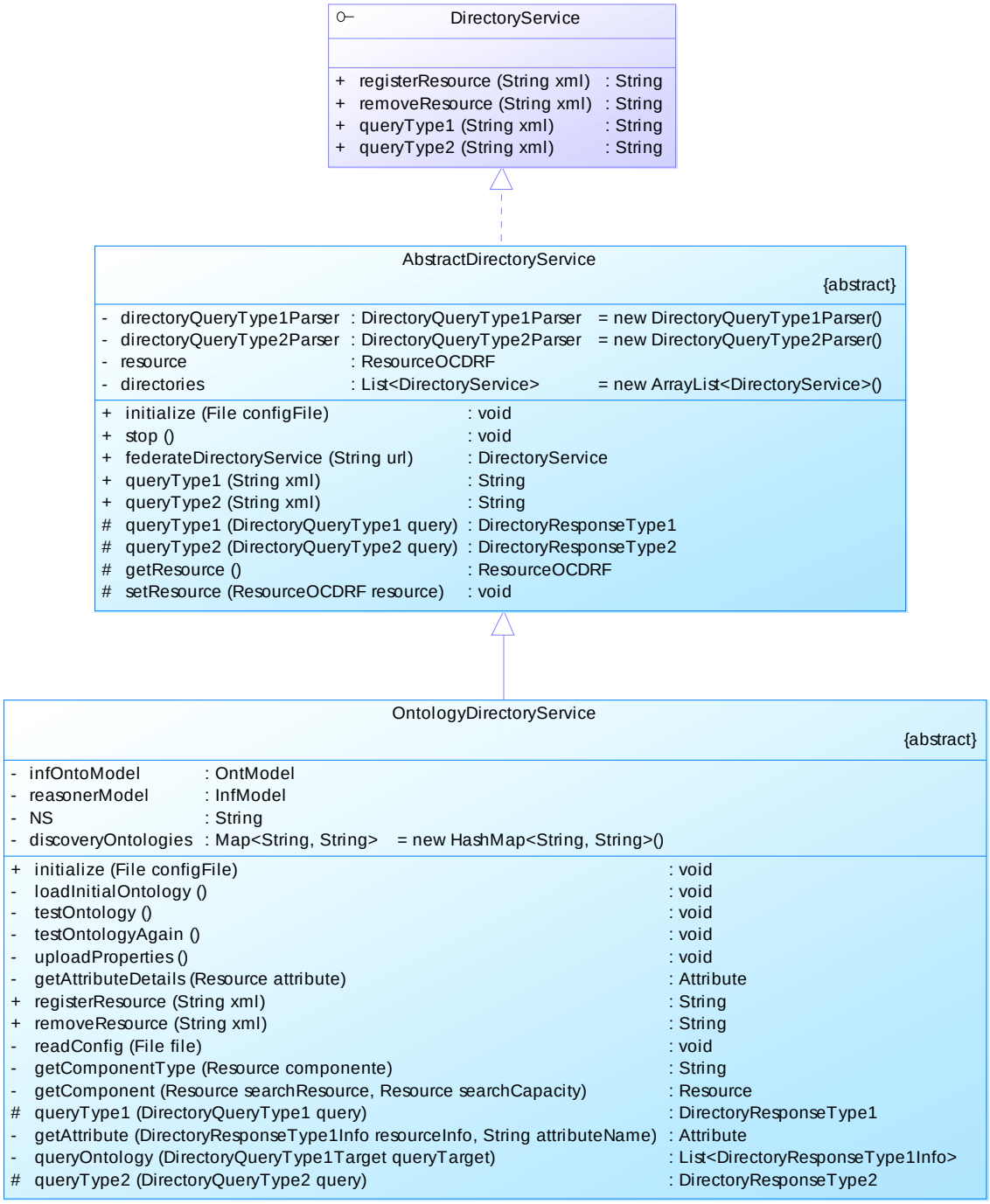

Figura 6 - Estrutura de classes do SRD

A classe abstrata *OntologyDirectoryService* é a classe responsável pela manipulação da Ontologia de Recursos. Ela possui o atributo *infOntoModel*, objeto da classe *OntModel*, disponibilizada pelo *framework* Jena. Este atributo encapsula a descrição dos componentes de um modelo ontológico e a associa a uma forma de armazenamento e a uma máquina de inferências. Então, *infOntoModel* representa o modelo ontológico inicial definido pela Ontologia de Recursos, associado a uma especificação de modelo ontológico pré-definida no

Jena, *OWL_MEM_MICRO_RULE_INF*, que permite que sejam feitas inferências considerando as propriedades de transitividade, inversão e simetria, além de fornecer bom desempenho.

O atributo *reasonerModel* é um objeto da classe *InfModel* do Jena. Esta classe permite a associação do modelo ontológico inicial às regras de inferência associadas à Ontologia de Recursos, estabelecidas como no arquivo ResourceRules.txt, configurado no OCDRF como apresentado no Código 28.

A classe *Resource* fornecida pelo Jena representa qualquer instância da ontologia. Ela é utilizada no OCDRF, para várias funções, entre elas, como parâmetro de entrada do método *getAttributeDetails* na classe *OntologyDirectoryService* que recupera as informações associadas às instâncias da classe *Attribute* referentes a um recurso.

O Serviço de Registro e Diretório é inicializado através do método *initialize*, que sobrescreve o método de mesmo nome de sua super classe conforme o Código 30.

```
 1 @Override
 2 public void initialize(File configFile) throws DirectoryServiceException
 3 {
 4     if (configFile.exists()) {
 5           readConfig(configFile);
 6           loadInitialOntology();
 7     } else {
 8           System.err.println("Configuration file not found: " +
configFile.toString());
 9           System.exit(4);
10     }
11 }
```

Código 30 - Inicialização do Serviço de Registro e Diretório

O método *initialize*, apresentado no Código 30, executa a leitura do arquivo de configuração do serviço através do método local privado *readConfig* (Linha 5) e a carga e verificação da Ontologia de Recursos inicial através do método *loadInitialOntology* (Linha 6). O método *readConfig* (Linha 5) é responsável por ler, além das configurações existentes em (RODRIGUES, 2009a), uma nova propriedade incluída no arquivo de configuração do SRD. Esta propriedade possui o nome do arquivo que contém a Ontologia de Recursos definida pelo administrador da infraestrutura e as regras de inferência que devem ser aplicadas a ela.

Depois de capturado o nome deste arquivo, o método *loadInitialOntology* é executado (Linha 6). Este método realiza a leitura da ontologia contida no arquivo e, com a utilização do

framework Jena (JENA, 2011), carrega classes Java que representarão as classes e as propriedades sobre estas classes que foram definidas na ontologia.

O arquivo que contém a Ontologia de Recursos e o arquivo que contém as regras associadas a esta ontologia são definidos no arquivo de configuração do SRD, conforme apresentado no Código 28.

As operações expostas pelo Serviço de Registro e Diretório são definidas pela interface *DirectoryService* apresentada no Código 31.

```
1  public interface DirectoryService {
2   String registerResource(String xml);
3   String removeResource(String xml);
4   String queryType1(String xml);
5   String queryType2(String xml);
6  }
```

Código 31 - Interface do Serviço de Registro e Diretório

A interface do SRD foi simplificada em comparação com a proposta em (RODRIGUES, 2009a). O SRD disponibiliza operações para registro de novos recursos "*registerResource*" (Linha 2) e para exclusão de um recurso do diretório "*removeResource*" (Linha 3). Além de operações de busca, "*queryType1*" (Linha 4), método principal de busca na Ontologia de Recursos, e "*queryType2*" (Linha 5), método que dá suporte aos pontos de extensão para buscas de propriedades definidas em Ontologias Cliente, que são as consultas estendidas apresentadas na Seção 2.4.1. Todas estas operações disponibilizadas pelo SRD recebem como parâmetro uma *string* XML que especifica os detalhes necessários para sua execução.

O método *registerResource*, implementado na classe *OntologyDirectoryService*, é o responsável por tratar os pedidos de registro de ARs e agregar à ontologia principal as configurações recebidas. Ao utilizar o método *registerResource,* os recursos que desejarem fazer parte da infraestrutura não precisarão obedecer a um formato fixo de mensagem, tampouco estar associados a tipos pré-cadastrados, como proposto em (RODRIGUES, 2009a). Não há necessidade de registro prévio com os tipos aceitos.

Um AR ao se registrar, informa suas características utilizando a linguagem OWL. Para isto devem ser definidas instâncias de classes da Ontologia de Recursos e novas classes e propriedades, caso seja necessário, assim como apresentado no exemplo do Código 39. Estas

informações são então adicionadas à Ontologia de Recursos já registrada no SRD. Para isto, o método *registerResource* faz a leitura do OWL enviado pelos ARs e o carrega no SRD utilizando o método *add* da classe *InfModel* do *framework* Jena.

Sempre que as informações de registro de um novo recurso são carregadas, o atributo privado *infOntoModel* da classe *OntologyDirectoryService* é atualizado. Desta forma, informações completas e atualizadas estão sempre disponíveis para as consultas efetuadas no SRD. Este trabalho é realizado pelo método *registerSubOntology*.

Na Figura 7 é apresentado o diagrama de sequência com as operações realizadas durante o processo de registro de um AR no sistema. Através do método *registerSubOntology*, a cada registro efetuado, novos elementos são adicionados à ontologia principal.

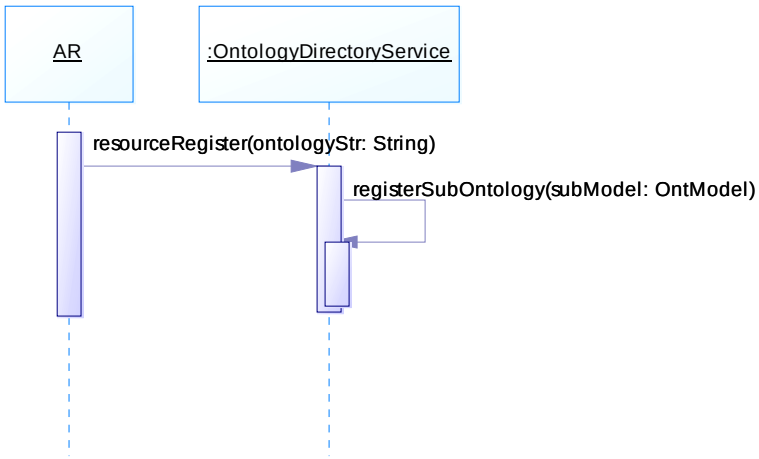

Figura 7 - Diagrama de Sequência da operação de registro de um AR

Os serviços da infraestrutura (Serviço de Contexto e o Serviço de Descoberta) também são tratados pelo SRD como recursos logo, também precisam se registrar enviando suas informações em formato OWL, assim como descrito nas Seções 4.3 e 4.4.

As buscas realizadas pelo OCDRF, através dos métodos "*queryType1*" e "*queryType2*", retornam as informações dos recursos que possuam características semanticamente equivalentes às pesquisadas. Para a análise dos recursos durante as buscas, são consideradas informações alcançadas através de inferências considerando todas as relações entre as instâncias da ontologia. Entre elas, relações especificadas nativamente na linguagem OWL como herança, e relações de equivalência de instâncias, "*sameAs*" e de classes, "*equivalentClass*".

O SD e SC realizam as consultas ao SRD através do método *queryType1* da interface *DirectoryService* que é implementado na classe *OntologyDirectoryService*. Este método invoca os métodos de consulta, disponibilizados pelo Jena, sobre o atributo *reasonerModel* para montar a lista de recursos de resposta. A partir desta lista, o *parser* implementado utilizando o pacote *com.thoughtworks.xstream* (XSTREAM, 2011) é utilizado para montar a mensagem de retorno no formato esperado.

Finalmente, o método *queryType2* pode ser invocado para tratar restrições de suas consultas do tipo *ClientConstraint*. Na implementação de referência, apenas o SD invoca o método *queryType2* disponibilizado pelo SRD. Este método recupera valores de propriedades definidas em Ontologias Cliente e atribuídas às instâncias da Ontologia de Recursos. As propriedades utilizadas para estas consultas devem estar corretamente configuradas na inicialização do SRD como apresentado no Código 6.

4.2.1 Representação dos Elementos da Infraestrutura

Na proposta deste trabalho, os próprios elementos da arquitetura do OCDRF são também representados e posteriormente localizados a partir da Ontologia de Recursos.

Os elementos do OCDRF poderiam ser mapeados na Ontologia de Recursos de várias formas. Uma forma seria criar uma instância de *Resource* que representasse o OCDRF, isto é que representasse um conjunto de serviços da infraestrutura. Nesta representação, os serviços SC e SD seriam mapeados como *ResourceComponents*.

Porém a forma adotada foi representar cada elemento da infraestrutura como um *Resource*. Desta forma, cada serviço pode ter sua implementação como um *web service* separado, com *string* de conexão própria. Isto permite uma maior escalabilidade e facilidade de migração do ambiente de execução de cada serviço. Um exemplo da representação utilizada para um serviço da infraestrutura é apresentado no Código 33 da Seção 4.3.

4.3 **Implementação do Serviço de Contexto**

O Serviço de Contexto sofreu alterações para que pudesse (i) interagir com o novo SRD, (ii) suportar buscas que realizam análise semântica dos termos, (iii) retornar suas respostas para as aplicações clientes utilizando os termos solicitados por estas e (iv) usar a

nova representação de recursos para que possa realizar a consulta do contexto atualizado dos diversos componentes dos recursos.

O Serviço de Contexto é representado na ontologia como um *Resource* e deve ser registrado no SRD durante a sua inicialização. O Serviço de Contexto é inicializado através do método *initialize* apresentado no Código 32. Este método executa a leitura do arquivo de configuração do serviço através do método local privado *readConfig* (Linha 3) e serializa a ontologia que contém suas características, apresentada no Código 33, através do método *serializeOntology* (Linha 4).

```
1 public void initialize(File configFile) throws ContextException
2 {
3      readConfig(file);
4      String ontologyStr = resource.serializeOntology();
5      registerContextService(ontologyStr);
6 }
```

Código 32 - Inicialização do Serviço de Contexto

O método *registerContextService*, Linha 5 do Código 32, executa o método *registerResource* disponibilizado pelo *DirectoryService*, passando como parâmetro sua descrição serializada.

O Código 33 apresenta a descrição do Serviço de Contexto que é enviada para o SRD.

```
 1 <?xml version="1.0"?>
 2 <!DOCTYPE owl [
 3     <!ENTITY res  "http://localhost/ocdrf/conf/Resource.owl#" >
 4     <!ENTITY owl  "http://www.w3.org/2002/07/owl#" >
 5     <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#" >
 6   ]>

 7 <rdf:RDF
 8  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 9  xmlns:rdfs= "http://www.w3.org/2000/01/rdf-schema#"
10  xmlns:owl = "&owl;"
11  xmlns:xsd = "&xsd;"
12  xml:base  = "&res;"
13  xmlns     = "&res;">

14 <owl:Ontology rdf:about="">
15  <rdfs:comment>CS Description</rdfs:comment>
16 </owl:Ontology>

17 <Resource rdf:ID="ContextService01">
18  <hasName rdf:datatype="&xsd;string">Context Service 01</hasName>
19  <hasComponent rdf:resource="#ContextService01Engine" />
```

```
20  <hasVendor rdf:datatype="&xsd;string">OCDRF</hasVendor>
21  <invokedBy rdf:resource="#CS01WS" />
22 </Resource>

23 <Sensor rdf:ID="ContextService01Engine">
24  <hasName rdf:datatype="&xsd;string">Context Service 01 Engine</hasName>
25  <hasCapacity rdf:resource="#ContextService" />
26 </Sensor>

27 <InvokeMethod rdf:ID="CS01WS">
28  <invokeString rdf:datatype="&xsd;string">
       http://localhost:1975/context/ContextService?wsdl</invokeString>
29  <hasTechnology    rdf:resource="#webservice" />
30 </InvokeMethod>

31 <InputCapacity rdf:ID="ContextService" />

32 </rdf:RDF>
```

Código 33 - Informações para registro do SC

É através da leitura das informações contidas no Código 33 que o SRD poderá localizar o SC e retornar suas informações para o Serviço de Descoberta durante a sua iniciação, que é detalhada na Figura 9 da Seção 4.4. A Figura 8 apresenta, em forma de diagrama, como o SC foi mapeado na Ontologia de Recursos. O SD foi mapeado de forma análoga.

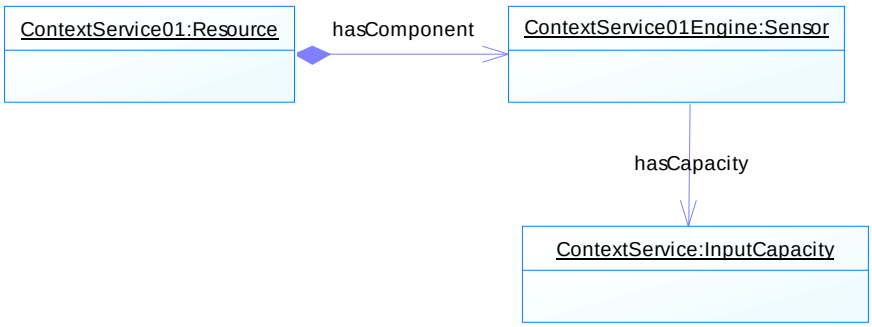

Figura 8 - Diagrama de mapeamento do SC na Ontologia de Recursos

Cada componente (sensor ou atuador) de um recurso deve possuir uma implementação fornecida por uma operação de um AR. Estas são representadas na Ontologia de Recursos através da classe *InvokeOperation* (Figura 4).

A implementação dos sensores de um recurso deve ser representada por uma operação que possui um nome, "*operationName*", e um texto como retorno obrigatório, sem possuir

parâmetros de entrada, com apresentado na Figura 5. Para que o Serviço de Contexto seja capaz de acessar as informações de contexto atualizadas de cada sensor de um recurso, ele deve ter acesso ao nome determinado para a operação a ser invocada, além de ter acesso à *string* de conexão para o AR.

As informações sobre a operação que implementa cada um dos componentes de um recurso são retornadas pelo SRD, de forma que o SC seja capaz de acessar a operação correta de cada AR para capturar as informações mais atualizadas dos seus sensores. De posse da operação correta a ser executada, o SC chama a operação adequada do AR como apresentado no Código 34.

```
1  private ResourceState getResourceState(String resourceURL, String operationName)
2      ResourceAgent agent = getResourceAgent(resourceURL);
3      String agentResponse = agent.getResourceState(operationName);
       ...
4  }
```

Código 34 - Fragmento do método *getResourceState* da classe *AbstractContextService*

A implementação de referência trata da execução de métodos associados aos sensores apenas. Nela o método *getResourceState* da interface *ResourceAgent* é chamado recebendo como parâmetro o nome da operação que deve ser executada. Cabe ao AR executar a operação correta através do uso de reflexão (ORACLE1, 2011) ou de outro mecanismo de programação. O Código 35 da Seção 4.5 apresenta um exemplo de definição de diferentes operações para os sensores disponíveis em um recurso. A *string* retornada após a consulta ao AR, Linha 3 do Código 34, deve então sofrer o *parsing* para que seja retornado um objeto do tipo *ResourceState*, para tratamento posterior no SC.

4.4 Implementação do Serviço de Descoberta

O Serviço de Descoberta sofreu alterações para (i) suportar os novos tipos de buscas oferecidos pelo OCDRF que realizam análise semântica dos termos, (ii) retornar suas respostas para as aplicações clientes utilizando os termos solicitados por estas e (iii) interagir com o novo SRD e novo SC.

O Serviço de Descoberta, de forma análoga ao Serviço de Contexto, é representado na ontologia como um recurso e também deve ser registrado no SRD durante a sua inicialização.

A Figura 9 apresenta o diagrama de sequência da implementação da inicialização do SD que se registra junto ao SRD e busca pelo SC no mesmo. O SD deve ser inicializado sempre após a inicialização e registro do SC no SRD.

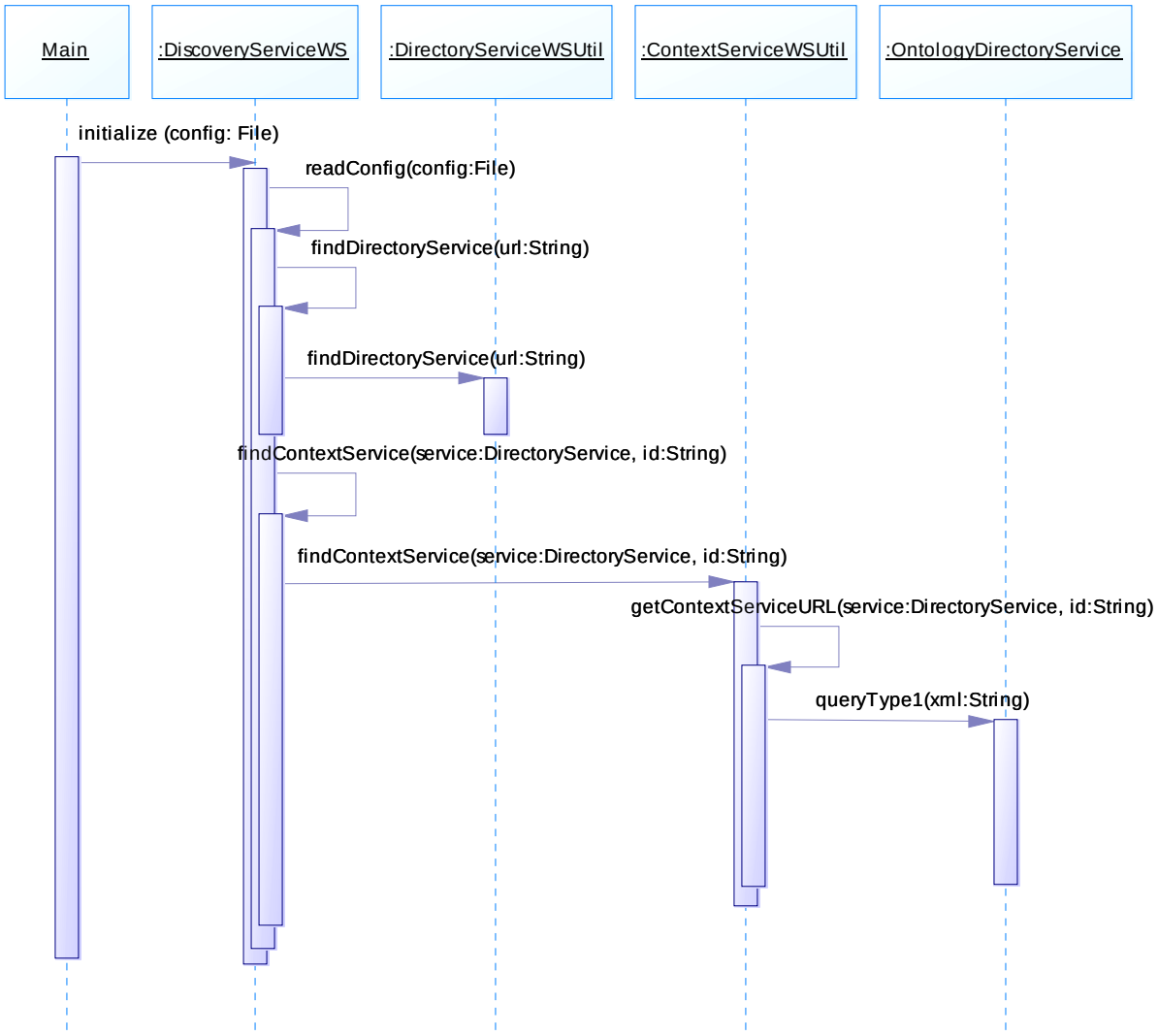

Figura 9 - Diagrama de Sequência da inicialização do SD

O Serviço de Descoberta, ao ser inicializado, efetua a leitura do seu arquivo de configuração para obter as informações sobre o SRD e a identificação do SC. A partir daí o SD é capaz de se conectar ao SRD e solicitar as informações de conexão para com o Serviço de Contexto. Para localizar as informações do SC, armazenadas na ontologia, o método *queryType1* disponibilizado pelo SRD precisa ser acionado.

Ao receber uma consulta, no formato especificado pelo Código 13, o SD inicia a análise sobre as restrições recebidas. Todas as restrições sobre a consulta devem ser avaliadas

como verdadeiras para que um recurso seja retornado para a aplicação cliente. Inicialmente, o SD analisa as restrições recebidas sobre as características das capacidades dos recursos (*CapacityConstraints)* e em seguida analisa as restrições que envolvem relações estabelecidas nas Ontologias Clientes (*ClientConstraints).* A Figura 10 apresenta os passos efetuados para esta pesquisa.

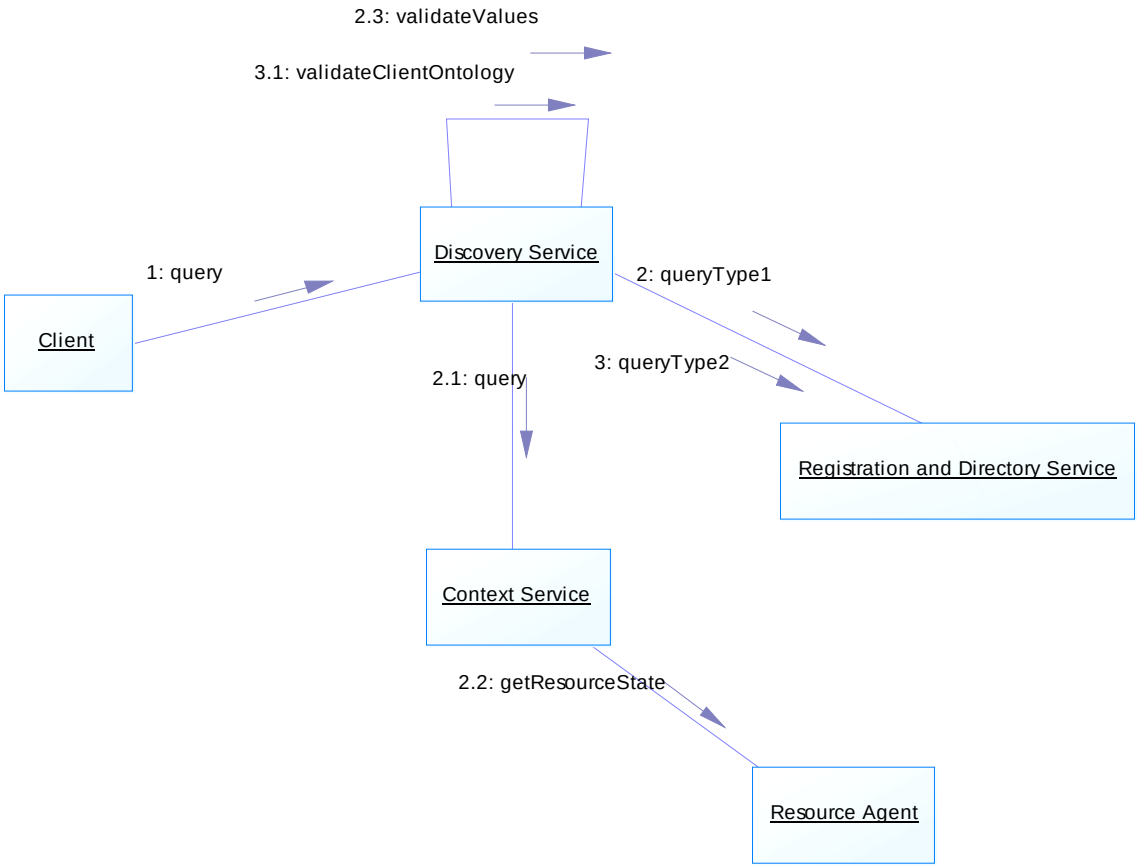

Figura 10 - Diagrama de Sequência da operação de busca do SD

Para cada *CapacityConstraint* encontrada na consulta, o SD verifica junto ao SRD pela existência do componente, capacidade e atributos buscados. O SRD efetua a busca pelos termos solicitados considerando a semântica dos mesmos e retorna os atributos encontrados informando os termos com os quais foram registrados, seu tipo, que indica se são atributos estáticos ou dinâmicos, o valor associado a estes atributos caso sejam estáticos e a unidade no qual os valores estão representados.

Após o retorno do SRD, o SD envia uma consulta para o SC com o objetivo de obter os valores atualizados dos atributos dinâmicos envolvidos na consulta da aplicação cliente. O SD envia para o SC os nomes dos atributos conforme eles foram registrados no SRD, de

forma que o SC possa consultar estes atributos diretamente nos ARs, sem que seja necessária nova consulta ao SRD para obter estes termos.

De posse das informações obtidas pelo SC, o SD inicia o processo de comparação dos valores retornados com os valores desejados pela aplicação cliente, observando a unidade em que cada um destes valores está representado e a operação de comparação desejada pela aplicação cliente. Caso os valores, estáticos e dinâmicos, obtidos estejam de acordo com os solicitados, o SD inclui o recurso na lista de retorno para a aplicação cliente e prossegue para a análise da próxima restrição sobre as capacidades do recurso (*CapacityConstraint*) a ser atendida. Esta sofre o mesmo tratamento realizado para a anterior.

Caso algum dos valores dinâmicos obtidos do SC ou dos valores estáticos obtidos do SRD não esteja de acordo com os solicitados, o SD paralisa a sua análise e retorna para a aplicação cliente uma lista de recursos vazia. Caso algum atributo não seja encontrado pelo SRD, o resultado desta busca é vazio e o SD também retorna para a aplicação cliente uma lista vazia.

Em caso de sucesso, após a análise das restrições sobre as capacidades do recurso (*CapacityConstraint*), o SD inicia a análise sobre as restrições que envolvem as relações estabelecidas nas Ontologias Clientes (*ClientConstraints*). Estas restrições vão se utilizar dos ganchos disponibilizados pelo OCDRF, representados na Figura 4 como “*ResourceHook*” e “*AttributeHook*”. O objeto de comparação utilizado para a análise deste segundo tipo de restrição é enviado na consulta, como definido pela Linha 28 do Código 13 (Seção 2.4.2).

O SD utiliza as classes *OntModel* e *InfModel* providas pelo Jena para realizar os casamentos semânticos com base na Ontologia Cliente. Estes casamentos semânticos levam em consideração a operação de comparação desejada, definida pela Linha 27 do Código 13, que deve representar uma propriedade definida dentro da própria Ontologia Cliente, como o exemplo no Código 38 (Seção 5.1.3.6).

Após o término da análise sobre os dois tipos de restrições solicitadas pela aplicação cliente, o SD finalmente se encarrega de formatar a lista de retorno para a aplicação cliente com os recursos encontrados, seguindo a especificação determinada pelo Código 9 (Seção 2.4.1).

4.5 Implementação de Agentes de Recursos

A nova modelagem de recursos através da Ontologia de Recursos permite que um recurso possua mais de um sensor ou atuador e que os mesmos sejam executados de forma independente conforme a necessidade das aplicações clientes. O Código 35 apresenta um exemplo de utilização de tal modelagem.

```
 1 ...

 2 <Resource rdf:ID="Telcomed_WristClinic_AIO67">
 3   <hasComponent   rdf:resource="#Telcomed_000000B8" />
 4   <hasComponent   rdf:resource="#Telcomed_000000DE" />
 5   <hasComponent   rdf:resource="#Telcomed_00BD0001" />
 6   <hasComponent   rdf:resource="#Telcomed_000000B9" />
 7   <hasComponent   rdf:resource="#Telcomed_00000141" />
 8   <invokedBy      rdf:resource="#Telcomed_WristClinic_WS" />
     ...
 9 </Resource>

10 <Sensor rdf:ID="Telcomed_000000B8">
11   <hasName rdf:datatype="&xsd;string">WRISTCLINIC AIO67 - BLOOD PRESSURE AND HEART RATE
MEASUREMENT</hasName>
12   <hasCapacity           rdf:resource="#BloodPressure" />
13   <hasCapacity           rdf:resource="#HeartRate" />
14   <hasInvokeOperation    rdf:resource="#telcomed_000000B8_operation" />
15 </Sensor>

16 <InvokeOperation rdf:ID="telcomed_000000B8_operation">
17   <operationName rdf:datatype="&xsd;string">get_BP_HR_state</hasName>
18   <invokeReturnType      rdf:datatype="&xsd;string">String</invokeReturnType>
19 </InvokeOperation>

20 <InvokeMethod rdf:ID="Telcomed_WristClinic_WS">
21   <invokeString     rdf:datatype="&xsd;string">
      http://localhost:2007/TelcomedWristClinic?wsdl</invokeString>
22   <hasTechnology    rdf:resource="#webservice" />
23 </InvokeMethod>

24 <InputCapacity rdf:ID="BloodPressure">
25   <hasAttribute   rdf:resource="#systolic" />
26   <hasAttribute   rdf:resource="#diastolic" />
27 </InputCapacity>

28 <InputCapacity rdf:ID="HeartRate">
29   <hasAttribute   rdf:resource="#pulse" />
30 </InputCapacity>

31 ...
```

Código 35 - Exemplo de definição de um Recurso

O Código 35 apresenta um exemplo de recurso composto por vários sensores. O *WristClinic* (TELCOMED1, 2011) possui a capacidade de realizar as seguintes medidas corpóreas: pressão arterial, frequência cardíaca, ECG, frequência respiratória, SPO2, além da temperatura. Ao ser corretamente posicionado no pulso do paciente e acionado, o *WristClinic* realiza mediçõess e envia os valores observados para um dispositivo acoplado à porta USB de um computador. Este recurso possui várias capacidades de medições, que podem ser realizadas de forma independente.

As Linhas 3-7 do Código 35 representam os cinco componentes disponibilizados pelo *WristClinic*. Cada um deles têm as suas capacidades detalhadas na ontologia, como nas Linhas 24-27 e 28-30, relacionadas ao sensor de identificação "*Telcomed_000000B8*", descrito nas Linhas 10-15.

Cada componente do recurso deve possuir a definição da operação que o implementa. As Linhas 16-19 apresentam a descrição da operação que implementa o sensor "*Telcomed_000000B8*". A operação "*telcomed_000000B8_operation*" é associada a ele na Linha 14. Esta operação faz parte do conjunto de operações disponibilizadas através do *web service* definido nas Linhas 20-23 que é associado ao recurso na Linha 8.

A implementação dos Agentes de Recursos sofreu alterações para suportar as diversas operações que podem estar disponíveis através do mesmo. Cada componente de um recurso deve possuir uma implementação fornecida por uma operação diferente dentro de um AR. Estas podem possuir os mais diversos nomes e parâmetros de entrada e saída.

A nova modelagem dos Agentes de Recursos suporta então que os mesmos representem as diversas operações fornecidas por seus sensores e atuadores, mas além desta nova característica, os ARs precisam possuir uma interface unificada para que sejam acessados através do Serviço de Contexto. Por isto optou-se pela utilização de um método unificado para extração das informações dos sensores de um recurso.

Utilizando o método unificado, as operações descritas na ontologia para cada sensor de um recurso podem ser disponibilizadas através do uso da API de reflexão provida pela linguagem Java (ORACLE1, 2011) no método *getResourceState* dos ARs. Desta forma a interface disponibilizada pelos ARs permanece única, porém provendo suporte às diversas operações especificadas. A interface *ResourceAgent* foi alterada para isto e é apresentada no Código 36.

```
1  public interface ResourceAgent {
2      String getResourceState(String operationName);
3  }
```

Código 36 - Interface de um Agente de Recursos

O método *getResourceState* passa a receber um parâmetro do tipo texto contendo o nome da operação a ser executada com o objetivo de extrair os dados mais atualizados do sensor no qual a aplicação está interessada. A implementação do método *getResourceState* pode então utilizar-se do mecanismo de reflexão disponível na linguagem Java ou de outro mecanismo de programação desejado, para executar a operação esperada, de posse do seu nome.

4.6 **Avaliação de Desempenho**

Para a avaliação de desempenho da infraestrutura proposta foi desenvolvido um programa capaz de simular tipos diversos de consultas à infraestrutura considerando uma proporção de utilização para cada consulta, como é esperado que estas sejam executadas em uma aplicação real.

Utilizando o programa de verificação de desempenho criado é possível estressar condições próximas das reais e não apenas medir a capacidade da infraestrutura para responder de forma massiva a um único tipo de consulta, como é comumente realizado. O programa permite que seja medida a capacidade da infraestrutura para responder a vários tipos de consulta, enviados massivamente de forma arbitrária, porém respeitando a proporção indicada. O objetivo desta análise é demonstrar como a infraestrutura se comportaria em casos reais, porém extremos, com tráfego intenso.

O programa criado utilizando a linguagem Java SE versão 6, funciona da seguinte forma: de acordo com as proporções informadas para cada tipo de consulta a ser realizada, o programa inclui em um vetor informações que indicam a chamada de cada consulta. O tamanho do vetor a ser populado, é determinado pela quantidade de testes que se deseja executar.

Após popular o vetor, a aplicação embaralha as informações contidas nele, de forma a garantir a aleatoriedade na chamada das consultas, que pode ser observada em aplicações reais.

De acordo com o volume de consultas a serem executadas por segundo, a aplicação inicia então *threads* para tratar cada uma delas, as enviando para a infraestrutura e aguardando sua resposta.

A ferramenta de avaliação de desempenho não deve rodar na mesma máquina onde é executada a infraestrutura a ser testada, para não degradar seu funcionamento. A execução das *threads* abertas para os testes de carga podem influenciar na execução das próprias consultas.

A aplicação de verificação de desempenho foi então executada em um *notebook* Intel(R) Core(TM) i5 M560 2.67GHz, 4GB de memória e Windows 7 Professional. Enquanto os serviços da infraestrutura e os Agentes de Recurso consultados foram executados em uma segunda máquina com a mesma configuração.

Foram solicitadas 20.000 consultas, sendo 18 por segundo, de acordo com a seguinte proporção:

60% de consultas ao Serviço de Contexto verificando um atributo de um recurso e

40% de consultas ao Serviço de Descoberta, sendo destas:

- 32% consultas com 3 restrições, 2 delas envolvendo componentes (um deles verificando um atributo) e 1 restrição relacionada a uma propriedade de uma Ontologia Cliente;
- 4% consultas com 2 restrições, ambas envolvendo componentes (um deles verificando um atributo);
- 2% consultas com 1 restrição relacionada a uma propriedade de uma Ontologia Cliente;
- 2% consultas com 3 restrições, 2 delas envolvendo componentes (sem atributos) e 1 restrição relacionada a uma propriedade de uma Ontologia Cliente.

Para o protótipo desenvolvido neste trabalho e apresentado na Seção 5.1, considerando sua configuração simplifiada, o número de requisições executadas para este teste de desempenho (20.000), representaria dois anos de execução da aplicação.

As consultas enviadas para a análise de desempenho representaram outras consultas dentro da infraestrutura, cujo percentual real é representado através da Figura 11.

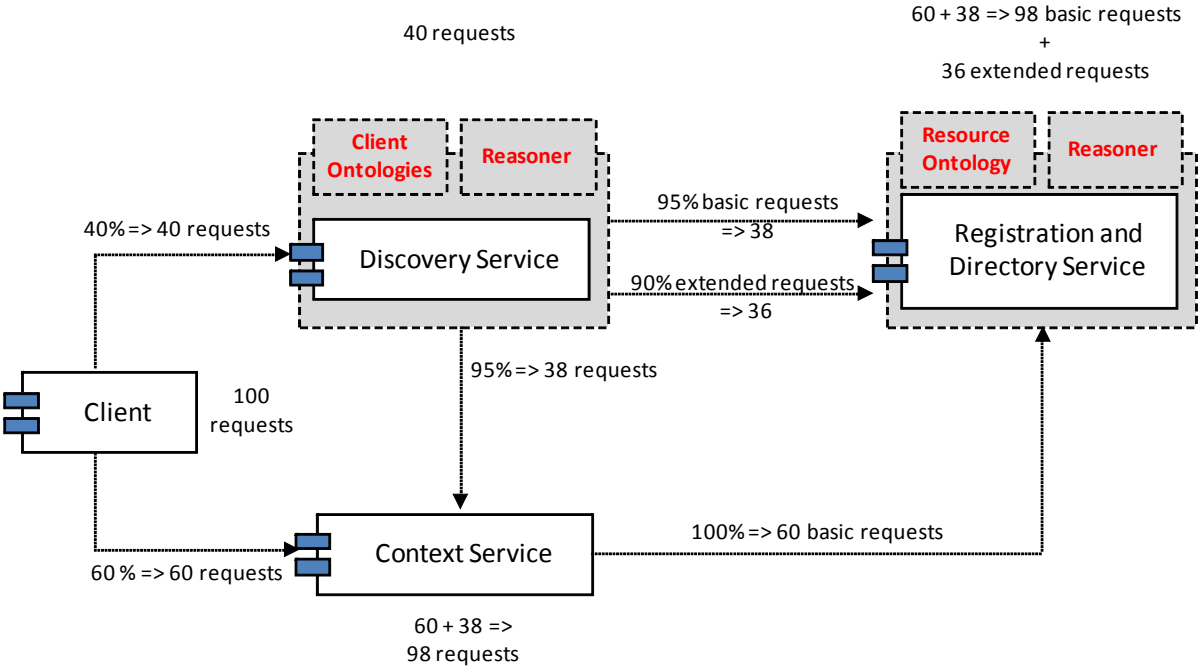

Figura 11 - Consultas efetuadas para análise de desempenho do OCDRF

A cada 100 requisições enviadas por parte de um cliente, segundo a expectativa utilizada para compor o teste de desempenho, 40 requisições serão recebidas pelo SD, 98 pelo SC e 134 pelo SRD, sendo destas últimas, 98 consultas que analisam somente a Ontologia de Recursos e 36 consultas que analisam Ontologias Cliente.

Para facilitar a leitura dos dados colhidos pela aplicação de verificação de desempenho, estes serão apresentados em gráficos separados por tipo de consulta. A Figura 12 apresenta o gráfico contendo os resultados apresentados pelo SC após a execução da aplicação de verificação de desempenho. As consultas efetuadas verificaram um atributo de um recurso.

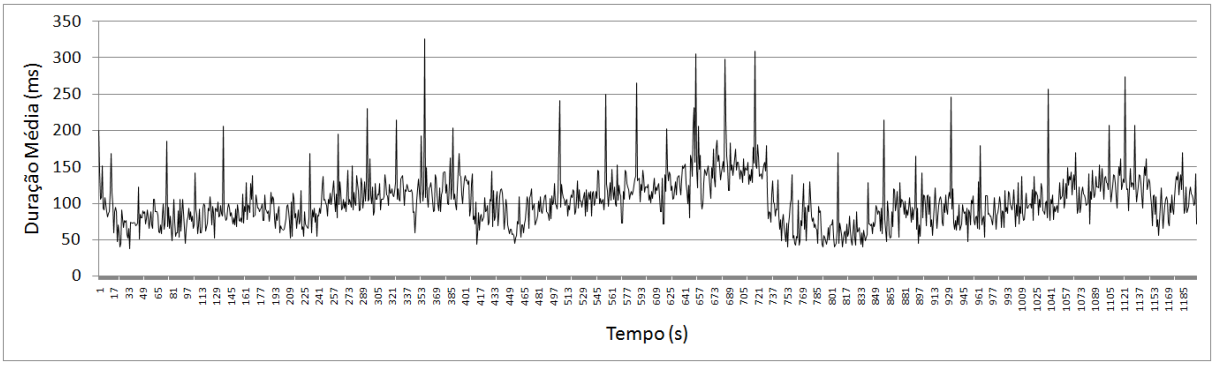

Figura 12 - Desempenho das consultas ao Serviço de Contexto

O eixo x da Figura 12, e dos próximos gráficos a serem apresentados, representa o tempo transcorrido em segundos e o eixo y representa a duração média das operações em milésimos de segundo.Vale observar que a quantidade de consultas ao longo dos testes foi fixa por segundo, uma vez que nenhuma consulta ultrapassou um segundo de duração.

As Figuras 13, 14, 15 e 16 apresentam os gráficos contendo os resultados apresentados pelo SD após a execução de cada tipo de consulta enviada pela aplicação de verificação de desempenho.

A Figura 13 apresenta o resultado das consultas efetuadas ao SD verificando três restrições, sendo duas sobre componentes (um deles verificando um atributo) e uma restrição sobre uma propriedade de uma Ontologia Cliente.

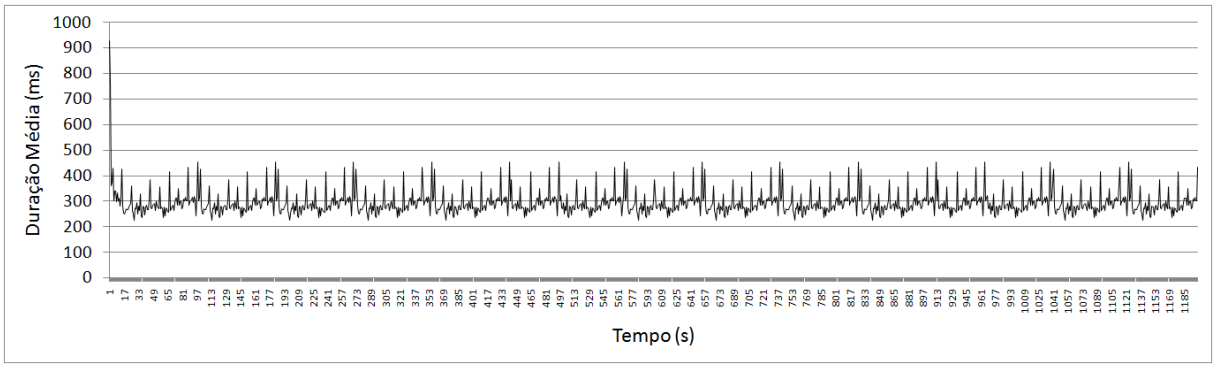

Figura 13 - Desempenho das consultas ao Serviço de Descoberta (3 restrições)

A Figura 14 apresenta o resultado das consultas efetuadas ao SD verificando duas restrições, ambas sobre componentes (um deles verificando um atributo).

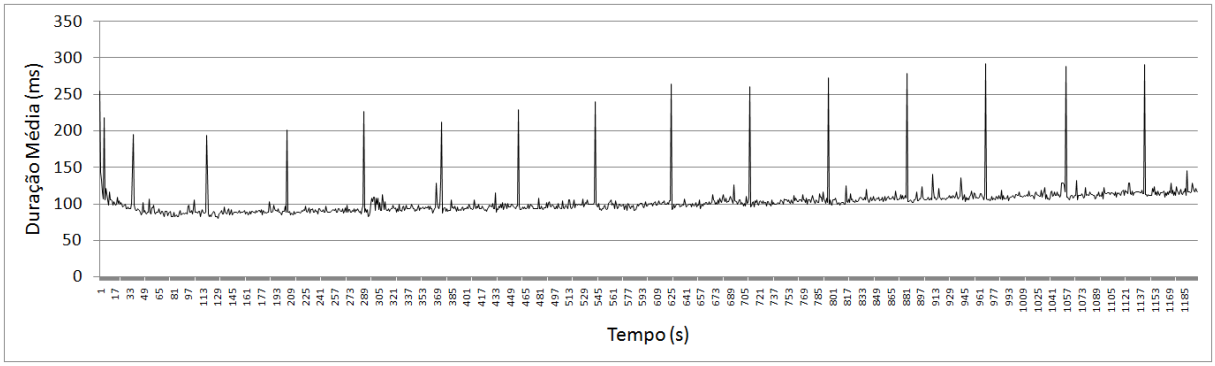

Figura 14 - Desempenho das consultas ao Serviço de Descoberta (2 restrições)

A Figura 15 apresenta o resultado das consultas efetuadas ao SD verificando uma restrição sobre uma propriedade de uma Ontologia Cliente. Vale observar que estas consultas não geram acesso ao SC.

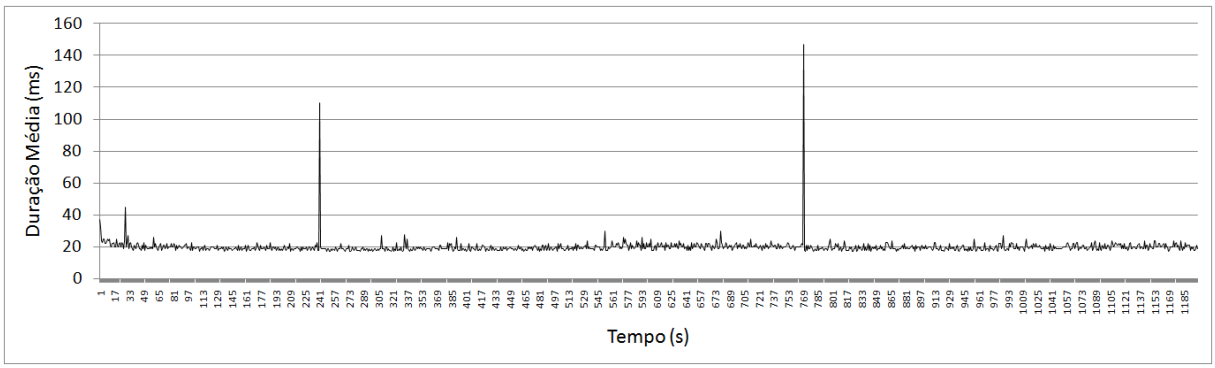

Figura 15 - Desempenho das consultas ao Serviço de Descoberta (1 restrição) – Sem SC

Para finalizar, a Figura 16 apresenta o resultado das consultas efetuadas ao SD verificando três restrições, duas delas sobre componentes (sem atributos) e uma sobre uma propriedade de uma Ontologia Cliente. Estas consultas não geram acesso ao SC.

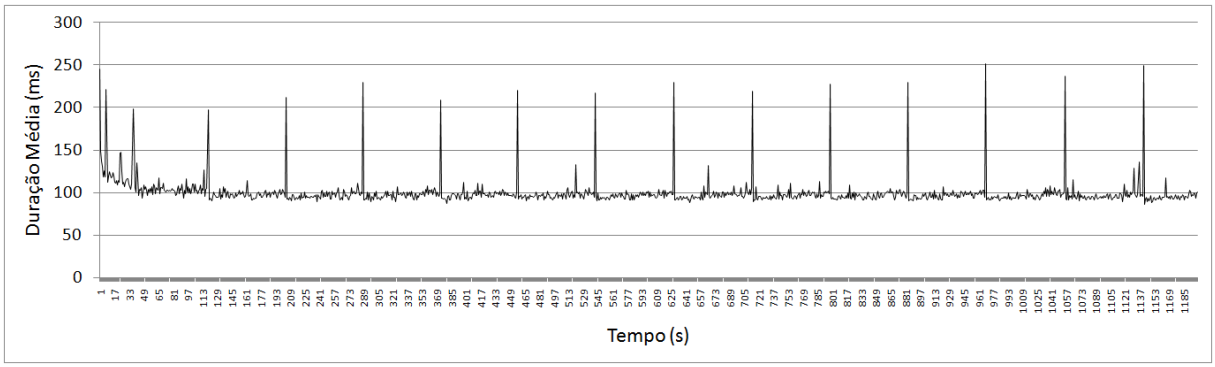

Figura 16 - Desempenho das consultas ao Serviço de Descoberta (3 restrições) – Sem SC

O comportamento da infraestrutura foi bastante linear ao longo da execução das consultas. O tempo médio de execução de consultas ao SC foi de 100ms e ao SD (considerando a consulta mais complexa) foi de 300 ms. No início dos testes, o tempo de resposta foi maior, porém este pode ser ignorado, pois rapidamente o sistema conseguiu se adaptar à carga exigida.

5 APLICAÇÕES DE EXEMPLO

Neste capítulo serão apresentadas aplicações cientes de contexto que utilizam a infraestrutura desenvolvida neste projeto. Tais aplicações utilizam as consultas semânticas e os pontos de extensão providos pela infraestrutura e demonstram a facilidade da utilização dos serviços providos e a flexibilidade na utilização de termos do domínio.

A primeira aplicação está no campo de telemedicina - *telecare/telehealth* (DLF, 2011), para a qual um protótipo foi desenvolvido.

A segunda aplicação oferece uma proposta para a seleção da melhor forma para acesso a dados em redes de telefonia celular e a terceira apresenta uma proposta para o controle de aquisição de suprimentos.

5.1 Aplicação em Telemedicina

5.1.1 Introdução

A telemedicina envolve equipamentos e serviços voltados à atenção médica a pacientes localizados à distância do centro médico. Ela engloba todo serviço que proveja saúde e cuidado diretamente ao usuário, muitas vezes em sua casa, suportado por tecnologias da informação e de telecomunicações. O emprego da telemedicina pode incluir por exemplo o monitoramento remoto do estilo de vida do paciente e dos seus sinais vitais, com a finalidade de diagnóstico e prevenção (DLF, 2011).

Através do cuidado domiciliar de saúde, a telemedicina visa dar aos pacientes o máximo de liberdade e qualidade de vida possível durante seu tratamento, fora do ambiente hospitalar. Estudos demonstram melhor resultado nos tratamentos e uma melhora na qualidade de vida dos pacientes (SLATER, 2006) (IDRISS, 2011) (DANG, 2009) quando estes permanecem em ambientes mais acolhedores, como em seu ambiente domiciliar. Porém para o sucesso do tratamento domiciliar é necessário que seja feito o monitoramento remoto destes pacientes, para que mesmo fora do ambiente hospitalar, estes sigam o tratamento necessário, de acordo com o estado de saúde em que se encontram.

Efetuar o tratamento de um paciente em sua casa gera diversos benefícios: menores custos, mais leitos disponíveis nos hospitais para os pacientes em estados mais graves ou com

indicação cirúrgica, e principalmente mais conforto, segurança e bem estar para o paciente e para sua família. Outro fator importante é a diminuição significativa do risco do paciente adquirir uma infecção hospitalar, resultando em redução da morbidade e mortalidade.

O contato mais próximo com sua família só traz benefícios ao paciente, que recupera sua privacidade e autoestima em um ambiente mais acolhedor e menos restritivo que o ambiente hospitalar. Isto se reflete diretamente na redução de atitudes negativistas (como agressividade e resistência ao tratamento) e na redução do tempo necessário para o tratamento. Este fato é expressivo principalmente na população pediátrica e geriátrica, que têm na internação domiciliar a manutenção do elo com a família em sua própria casa. Mas para que o tratamento em domicílio seja possível, em muitos casos, são necessários estrutura e suporte profissional adequados.

Muitas empresas do ramo de saúde já estão provendo serviços para tratamento em domicílio. Entre elas podemos citar as empresas brasileiras (PRONEP, 2011), (PROCARE, 2011), (MEDLAR, 2011) e (HOSPITAL-LAR, 2011) e muitas outras empresas nos Estados Unidos como a (GENTIVA, 2011) e no Canadá como a (KARPHOMECARE, 2011), que fazem o acompanhamento do paciente no seu dia a dia e atuam em situações de emergência. Devido à dificuldade de locomoção de muitos idosos e o eminente risco de quedas, estão disponíveis, por exemplo, serviços que provêem atendimento emergencial a pacientes em casos de quedas.

O sucesso de um tratamento médico depende em muitos casos do monitoramento constante do paciente. A efetividade e sucesso da telemedicina requerem então a possibilidade de verificação remota da conformidade com o tratamento estabelecido e das reações dos pacientes diante do tratamento. Isto demanda a obtenção, registro e análise das mais variadas informações sobre o dia a dia do paciente. Por exemplo, pode ser necessário verificar se o paciente toma os medicamentos indicados por seus médicos nos horários corretos, suas atividades ao longo do dia e suas medidas fisiológicas (e.g., pressão arterial).

Pacientes idosos comumente possuem problemas de memória e a possibilidade destes receberem avisos quanto às atividades que precisam cumprir é essencial para um tratamento adequado. Dependendo da enfermidade é preciso que o paciente também efetue medições de suas taxas fisiológicas com determinada frequência ao longo do dia. Todos estes exemplos expõem a necessidade de equipamentos e de uma infraestrutura de suporte adequada para viabilizar a telemedicina para uma grande gama de pacientes.

Atualmente estão disponíveis no mercado diversos tipos de equipamentos, de diferentes fornecedores, que podem ser utilizados para a monitoração das atividades de um paciente e para a verificação de suas medidas fisiológicas no seu ambiente domiciliar. Muitos destes equipamentos podem ser manuseados pelo próprio paciente, trazendo-lhe maior autonomia e segurança sobre seu estado de saúde. Entre eles está o *WristClinic* (TELCOMED1, 2011). Ele possui a capacidade de medir pressão arterial, frequência cardíaca, SpO2, entre outras medidas. Ao ser devidamente ajustado ao pulso e acionado, o *WristClinic* efetua a medição das taxas e envia os valores obtidos para um dispositivo denominado *MiniGate* (do mesmo fabricante), que deve estar acoplado à porta USB de um computador. Outro exemplo de equipamento médico que pode ser manuseado pelo próprio paciente é o *kit CP-1THW* da A&D (A&D, 2011), que contém um monitor de pressão sanguínea, uma balança e um monitor de atividades, todos sem fio, além de um USB *Transceiver*. Estes equipamentos podem fornecer automaticamente entradas para um sistema de monitoramento voltado para a telemedicina.

A telemedicina é então um campo que pode obter inúmeros benefícios com a utilização de aplicações sensíveis ao contexto, capazes de lidar com os mais diversos dispositivos disponíveis no ambiente domiciliar dos pacientes. A utilização do OCDRF para tais aplicações aumenta a agilidade de desenvolvimento das mesmas, por encapsular os aspectos de registro e descoberta de recursos, além da comunicação com os mesmos para extração de informações de contexto. Sua utilização também garante a localização de recursos através da análise semântica das consultas, conforme descrito no Capítulo 2.

Para demonstrar a utilização da infraestrutura desenvolvida neste trabalho para o suporte às aplicações voltadas para telemedicina é proposta uma aplicação com base no projeto SCIADS, descrito a seguir.

5.1.2 SCIADS

Através de sensores utilizados no ambiente domiciliar, um paciente pode ser monitorado a todo momento e em qualquer parte da casa. Dados fisiológicos (pressão arterial, frequência cardíaca, etc.), atividades realizadas pelo paciente (se está caminhando, dormindo, comendo, etc.) e condições do ambiente (temperatura e umidade) podem ser obtidos continuamente. Com base na captação, por meio do telemonitoramento, e cruzamento destas informações, profissionais de saúde como cuidadores, enfermeiros e médicos têm a

oportunidade de acompanhar remotamente o dia a dia do paciente e ajustar o seu plano de cuidados, de acordo com a evolução do tratamento.

Para este fim foi proposto o projeto SCIADS (Sistema Computacional Inteligente de Assistência Domiciliar à Saúde) (SCIADS, 2011) que integra, através de uma infraestrutura baseada em computação ciente de contexto, diversos aspectos relevantes ao monitoramento remoto da saúde do paciente em seu ambiente domiciliar.

Em (CARVALHO, 2010) foi proposto um protótipo cujo foco foi a contínua identificação da situação de saúde do paciente, associada à definição de um plano de cuidados. Para o monitoramento dos pacientes foram empregados dispositivos com comunicação sem fio, como um sensor de pressão arterial e frequência cardíaca, e sensores de movimento.

O sistema coleta dados fisiológicos e a atividade realizada pelo paciente e os analisa usando um modelo em lógica *fuzzy* (COPETTI, 2009), definido com base em regras produzidas em cooperação com especialistas médicos. Este módulo de análise é responsável por determinar a situação de saúde do paciente, definida em três faixas possíveis: normal, alerta ou emergência.

Para o gerenciamento e acesso aos dados de contexto atualizados dos dispositivos empregados, os serviços de Contexto e Descoberta providos pela infraestrutura CDRF podem ser aplicados assim como apresentado em (RODRIGUES, 2009b) e (RODRIGUES, 2009c). A utilização de tal infraestrutura tira da aplicação a responsabilidade de contemplar os aspectos específicos de comunicação com cada dispositivo, de diversos fabricantes, para a coleta de informações. Esta interação fica a cargo dos Agentes de Recurso e dos serviços citados, que encapsulam os mecanismos específicos de cada sensor e atuador do contexto de execução das aplicações, em módulos com interfaces padronizadas. Através da utilização desta infraestrutura, a interação pode ocorrer por meio de trocas de mensagens síncronas e assíncronas (eventos *push* ou *pull*).

Os experimentos do projeto SCIADS podem se beneficiar da análise semântica provida pelo OCDRF através da utilização de ontologias. A utilização de ontologias oferece mais flexibilidade e semântica na utilização dos termos do domínio e maior eficiência na busca pelos recursos do contexto de execução da aplicação, dado que será considerada a correlação semântica e não apenas sintática durante a execução de tais buscas.

5.1.2.1 Módulos do SCIADS

A aplicação se divide em dois módulos: (i) o módulo CLIENTE - aplicação de telemonitoramento que será executada em um terminal na residência do paciente e será reponsável por monitorar os eventos ocorridos neste ambiente; e (ii) o módulo CENTRAL – que será executado na central de saúde, onde profissionais da área de saúde devem monitorar remotamente os pacientes em tratamento e garantir um auxílio rápido em caso de necessidade de intervenção humana. A Figura 17 apresenta os módulos da aplicação de telemonitoramento.

A aplicação funciona de forma autônoma na casa do paciente, se comunicando com a aplicação executada na central de saúde para reportar (i) informações sobre a saúde do paciente, (ii) alarmes em caso de não cumprimento do Plano de Cuidados e (iii) alarmes em situações analisadas pela aplicação como emergências.

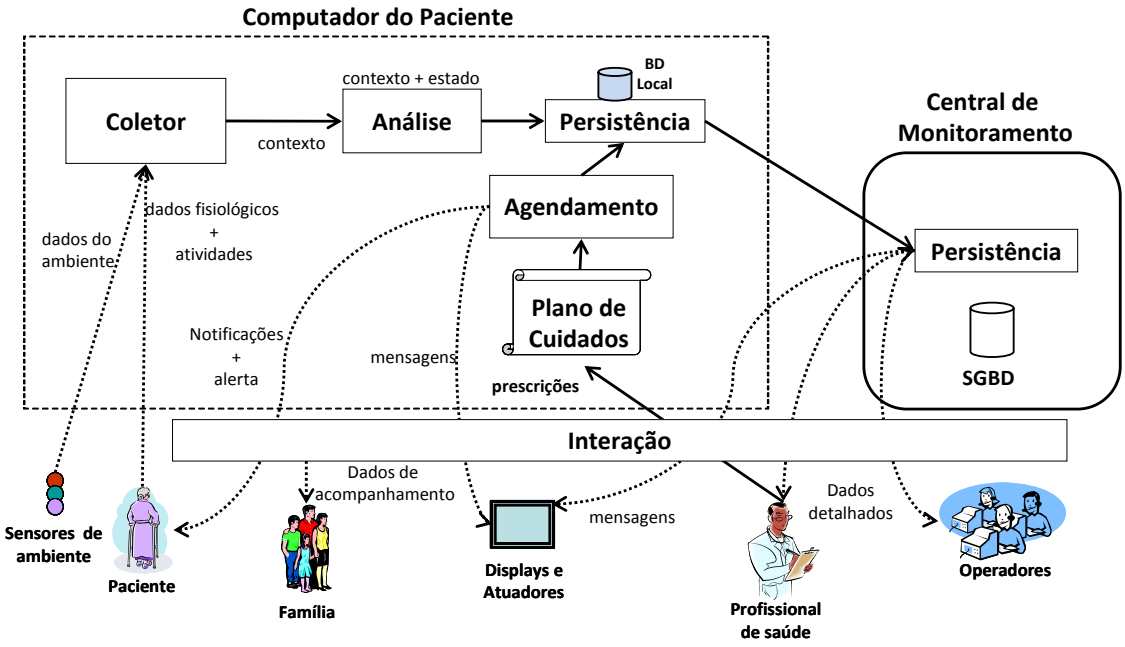

Figura 17 - Módulos do Sistema de Telemonitoramento

Os módulos de Plano de Cuidados, Coletor, Análise e Agendamento são executados em um computador localizado na residência do paciente, possibilitando o início de ações locais e a persistência dos dados coletados localmente por questões de redundância.

- Plano de Cuidados: Contém as atividades, cronograma de medições e ingestão de medicamentos prescritos pelo médico;
- Coletor: Monitora sensores e aparelhos para coletar dados de contexto;
- Análise: Aplica uma série de regras e utiliza um modelo de lógica *fuzzy* para inferir o estado do paciente, tendências de gravidade ou situações críticas. Este módulo deve ser configurado de acordo com a patologia associada ao paciente;
- Agendamento: Programado para enviar mensagens de alerta, transmitir dados de contexto para a Central de Monitoramento e disparar ações pré-configuradas em momentos específicos segundo o Plano de Cuidados recebido da Central.

Na Central de Monitoramento os dados dos vários pacientes (e respectivos dados de anamnese e dados de contexto), médicos e outros atores do sistema são persistidos em uma base de dados relacional.

- Persistência: No sistema local uma camada de persistência prepara os dados de contexto coletados para serem transmitidos e persistidos em uma base de dados na Central de Monitoramento. Na Central o módulo de persistência oferece os mecanismos necessários para o armazenamento e recuperação dos dados com independência do gerenciador específico;
- Interação: Controla a apresentação dos dados monitorados, fornecendo visões apropriadas para cada ator do sistema, ou seja, o paciente, membros da sua família e profissionais de saúde autorizados.

Médicos, pacientes, cuidadores e familiares podem ter acesso à visualização dos dados através da Internet, com as devidas credenciais. Médicos também podem acompanhar a evolução do paciente ou alterar o Plano de Cuidados através desta mesma interface.

Na central de monitoramento, os profissionais de saúde têm acesso aos prontuários clínicos dos pacientes. De posse das informações capturadas através dos sensores e Agentes de Recurso na casa de cada paciente, estes profissionais podem tomar as decisões apropriadas em cada situação (SZTAJNBERG, 2009).

5.1.3 Características da Aplicação

Nesta seção são discutidos os aspectos importantes da aplicação. Alguns destes aspectos, como o Plano de Cuidados foram abordados previamente no projeto, mas são aprofundados aqui, e outros, como a Cadeia de Acionamentos e a integração com o OCDRF são novas contribuições ao mesmo.

5.1.3.1 Plano de Cuidados

Uma aplicação de telemonitoramento pode permitir que médicos definam um Plano de Cuidados, como proposto em (CARVALHO, 2010), para cada paciente participante do programa de telemonitoramento. O Plano de Cuidados deve possuir as informações sobre as atividades obrigatórias ao longo do dia, para o tratamento do paciente.

É no Plano de Cuidados que estarão contidas as orientações quanto às medições fisiológicas que o paciente deve realizar, os equipamentos que devem ser utilizados para tais medições, os medicamentos a serem tomados e os horários de cada uma destas atividades, além de outras recomendações personalizadas conforme o tratamento.

Com base no Plano de Cuidados, a aplicação fica responsável por: (i) enviar avisos no horário estabelecido sobre a necessidade da execução de alguma atividade (que algum medicamento seja ministrado ou que alguma medição de uma de suas taxas fisiológicas seja realizada, por exemplo) e (ii) monitorar a execução de tal passo do Plano de Cuidados.

5.1.3.2 Cadeia de Acionamentos

O melhor meio para o envio de mensagens para o paciente pode depender das condições de saúde do mesmo. Este pode ser deficiente auditivo, por exemplo, sendo então necessário que as mensagens sempre sejam enviadas para ele por meio visual. Devido a isto a aplicação deve requerer a configuração de preferências da forma de envio e destino do aviso para cada tipo de item no Plano de Cuidados cadastrado no sistema.

A configuração dos múltiplos meios de envio e destinatários dos avisos é chamada de Cadeia de Acionamentos. Esta cadeia deve ser utilizada conforme o contexto do paciente: o ambiente, sua localização e o tempo de resposta para a execução do passo pendente.

A partir do envio do aviso, a aplicação deve aguardar uma interação do paciente para dar por concluído o passo do Plano de Cuidados. A aplicação deve então registrar os dados coletados e os enviá-los para uma central de saúde para acompanhamento.

Cada item no Plano de Cuidados que precise de intervenção por parte do paciente, profissionais de saúde ou familiares, deve ter associado a ele uma Cadeia de Acionamentos. Ela determina a forma de envio de avisos para o paciente ou para outras pessoas envolvidas, de acordo com o cumprimento ou não da tarefa solicitada após um intervalo de tempo determinado. A Cadeia de Acionamentos representa as preferências do paciente e de sua família e as exigências estabelecidas pela equipe de saúde.

Na Cadeia de Acionamentos é configurada a ordem de acionamento, a forma de contato e a temporização entre as mensagens. Quando uma tarefa do Plano de Cuidados é acionada, a aplicação busca a Cadeia de Acionamentos associada à mesma e, dentro dela o meio de envio de maior prioridade. Com esta informação a aplicação deve consultar o Serviço de Descoberta para localizar um dispositivo que esteja disponível para o envio do aviso. Uma vez localizado o dispositivo, a aplicação deve enviar o aviso para o destino cadastrado e iniciar a monitoração da execução da tarefa.

Caso uma tarefa do Plano de Cuidados não seja cumprida após o tempo determinado, apesar da solicitação enviada, a aplicação deve escalar a requisição de cumprimento da mesma. A aplicação deve então informar tal pendência para o próximo envolvido cadastrado, mais uma vez consultando o SD para localizar o recurso disponível segundo a forma cadastrada para o envio do aviso. O próximo envolvido cadastrado pode ser o próprio paciente, sendo contactado através de outro meio, como SMS por exemplo.

Conforme os temporizadores cadastrados na Cadeia de Acionamentos alcancem seus limites, novos envolvidos cadastrados devem ser acionados, o que pode incluir familiares, médicos e a central de monitoramento, de acordo com a condição de saúde de cada paciente e a urgência de que a ação pendente seja executada.

Caso não seja localizado o recurso necessário para o envio da mensagem, a aplicação deve buscar diretamente pela próxima configuração na Cadeia de Acionamentos, sem aguardar a temporização cadastrada. Por segurança, a aplicação deve possuir uma configuração padrão de envio de avisos para a central de monitoramento de saúde, para que o paciente não deixe de ser monitorado por falha na configuração da Cadeia de Acionamentos.

A Figura 18 apresenta um exemplo de Cadeia de Acionamentos para uma mensagem de lembrete de horário de medicamento, seguindo o Plano de Cuidados:

1) Aviso visual no ambiente em que o paciente se encontra – limite de tempo: 5 minutos
2) Aviso sonoro utilizando um dispositivo próximo ao paciente – limite de tempo: 5 minutos
3) SMS para o próprio paciente – limite de tempo: 5 minutos
4) Aviso no monitor da central de monitoramento – limite de tempo: 5 minutos
5) SMS para seu médico

Figura 18 - Exemplo de Cadeia de Acionamentos

5.1.3.3 Varredura de recursos

Durante a inicialização da aplicação uma varredura dos recursos registrados no sistema deve ser executada através da utilização do Serviço de Descoberta. A partir das informações obtidas devem ser realizadas validações sobre a disponibilidade de certos recursos com base no Plano de Cuidados. Desta forma, a aplicação é capaz de garantir que todos os recursos necessários para a execução dos passos do Plano de Cuidados estejam disponíveis.

Caso a aplicação não encontre um recurso com a capacidade necessária para a execução de um passo no Plano de Cuidados, ela deve informar tal problema para uma aplicação da Central de Monitoramento, cabendo à equipe médica tomar uma ação paliativa.

O módulo da aplicação na Central pode possuir o registro do caminho para os Serviços de Descoberta e Contexto ativados nas máquinas de todos os pacientes do programa de telemonitoramento, além do registro dos serviços da infraestrutura executados na própria central, e do cadastro do Plano de Cuidados de todos os pacientes. Assim este módulo é capaz de verificar por dispositivos disponíveis na central ou ainda na casa de pacientes, que não estejam em uso. Isto permite que um recurso seja realocado, garantindo melhor utilização dos recursos e consequentemente menores gastos para a operação do serviço de assitência médica.

5.1.3.4 Monitores de Emergência

Além do Plano de Cuidados, acionado com o decorrer do horário do sistema, a aplicação também pode fornecer a funcionalidade de Monitores de Emergência. Através deste mecanismo a aplicação é capaz de monitorar situações que estejam afetando ou possam afetar o estado de saúde do paciente como quedas e anormalidades na fisiologia do paciente, e acionar os envolvidos conforme configurado, de forma automática. Módulos de análise, como o proposto em (COPETTI, 2009) para análises de problemas com pressão arterial, podem ser usados pela aplicação para o disparo de uma emergência.

Um Monitor de Emergência também pode indicar uma situação de urgência apontada pelo próprio paciente. Este pode ter a seu dispor um dispositivo que, uma vez acionado, indique um estado de emergência ou pânico, como por exemplo, o *aLsis* da TelcoMed (TELCOMED2, 2011). Para isto um Monitor de Emergência, que monitore tal dispositivo, deve ser disponibilizado na aplicação. Tal dispositivo pode dar ao paciente maior confiança, permitindo-lhe acionar a equipe de suporte quando julgar necessário.

Mensagens dos Monitores de Emergência também devem ser associadas a uma Cadeia de Acionamentos. Caso a aplicação detecte uma situação de emergência, uma mensagem de alerta deve ser disparada. Esta terá um tratamento diferenciado, seguindo uma cadeia específica, podendo ser enviada primariamente para a equipe de suporte hospitalar e a posteriori para familiares, por exemplo.

5.1.3.5 Conclusão de um passo

Para que a temporização seja paralisada e a tarefa estabelecida no Plano de Cuidados seja marcada como executada ou o estado de emergência seja desabilitado é necessária uma interação por parte do paciente com a aplicação. Para isto, deve ser realizada a monitoração do estado de determinados recursos por parte da aplicação. Para cada item no Plano de Cuidados ou nos Monitores de Emergência deve ser configurado o tipo de recurso a ser monitorado como resposta, isto é, a capacidade provida pelos recursos a serem monitorados para a conclusão de um passo no Plano de Cuidados ou de um estado de emergência.

Para a monitoração do estado dos recursos desejados, o Serviço de Descoberta e o Serviço de Contexto providos pelo OCDRF devem ser utilizados. A aplicação deve descobrir,

utilizando o SD, os recursos disponíveis com tal capacidade e se registrar junto ao SC, como *Observer* dos mesmos, para receber suas informções de contexto atualizadas.

Caso a tarefa pendente exija uma entrada de informação no sistema, como uma medição fisiológica, as novas informações capturadas através do dispositivo de medição representam que o paciente fez o que lhe foi solicitado e a partir desta observação, a temporização deve ser desativada.

Caso a tarefa pendente seja um lembrete de uma atividade que não exija entradas específicas de informações no sistema, como um lembrete de medicação, o paciente precisa ter à sua disposição um recurso específico para realizar a marcação da finalização de tal pendência. Neste trabalho tal capacidade em um recurso foi denominada "*FeedbackButton*". Esta capacidade pode ser implementada utilizando o próprio computador onde a aplicação é executada na casa do paciente ou este pode possuir outro dispositivo próprio para isto, como por exemplo, o *aLsis* da TelcoMed (TELCOMED2, 2011).

5.1.3.6 Integração com o OCDRF

A aplicação apresentada pode utilizar o OCDRF para a identificação e comunicação com os recursos necessários para seu funcionamento, se valendo dos serviços semânticos providos por tal infraestrutura. Para isto, os recursos disponíveis na residência do paciente precisam ser registrados junto à infraestrutura, fornecendo sua descrição conforme a Ontologia de Recursos descrita na Seção 3.1.

Como exemplo de utilização da infraestrutura serão detalhados os passos necessários para o envio de mensagens para o paciente de acordo com a primeira e segunda formas de acionamento listadas na Figura 18.

Para envio de uma mensagem para o paciente faz-se necessário que a aplicação conheça a localização dos recursos que possui a seu dispor. Para isso, a descrição de cada recurso deve conter também sua localização, i.e., uma associação com um cômodo da residência.

Além disso, é necessário que a aplicação conheça a localização do próprio paciente. Para isto, um Agente de Recurso associado a um Sistema de Localização de Tempo Real (RTLS – *Real Time Location System*), como o provido pela Ekahau (EKAHAU, 2011) também deve ser utilizado.

A segunda forma de acionamento listada na Figura 18 utiliza ainda o relacionamento “próximo de”, diferente do clássico relacionamento de igualdade, entre a localização do paciente e a localização do recurso, para que a mensagem seja transmitida através de áudio.

Para a utilização de um relacionamento como o “próximo de” é necessário que seja realizado o mapeamento dos cômodos da residência do paciente e que seja estabelecida uma correlação de proximidade entre os mesmos. Para que este mapeamento possa então ser utilizado pelo OCDRF durante a análise das consultas recebidas, tal mapeamento deve ser realizado através de uma ontologia. Além disso, as propriedades desta ontologia que precisarem ser posteriormente utilizadas nas consultas ao Serviço de Descoberta, devem ser configuradas neste serviço conforme apresentado no Código 37.

A Ontologia de Localização definida para utilização pela aplicação de telemedicina deve incluir informações sobre a composição e a proximidade entre os cômodos da residência do paciente.

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <DiscoveryService>
     . . .
3    <DiscoveryOntologies>
4       <DiscoveryOntology Property="locatedIn" File="/conf/Location.owl" Rules="" />
5       <DiscoveryOntology Property="closeTo"   File="/conf/Location.owl" Rules="" />
6    </DiscoveryOntologies>
7 </DiscoveryService>
```

Código 37 - Exemplo de configuração de DiscoveryOntologies no discoveryService.xml

O Código 37 apresenta a configuração do SD para a utilização do ponto de extensão existente no OCDRF conforme discutido na Seção 3.2. Através deste ponto de extensão é possível utilizar propriedades de novas ontologias disponibilizadas pela aplicação cliente em consultas enviadas para o SD. Esta funcionalidade dá flexibilidade à aplicação cliente para utilizar os relacionamentos entre os termos da sua área de atuação da maneira que melhor lhe convier.

Desta forma, o OCDRF permite a utilização livre das terminologias utilizadas nas diversas áreas da telemedicina atráves dos pontos de extensão e utilização de ontologias, permitindo que desenvolvedores, profissionais de saúde e pacientes troquem informações de forma consistente.

De forma resumida, para utilizar os pontos de extensão do OCDRF e da Ontologia de Recursos, cada propriedade (relacionamento) a ser utilizada nas consultas da aplicação cliente precisa ser cadastrada na configuração do SD e SRD para que estes serviços saibam a qual ontologia cada propriedade se refere. Adicionalmente, as informações que devem ser comparadas através destas propriedades devem ter uma correlação com alguma propriedade na Ontologia de Recursos.

A propriedade "*locatedIn*", configurada na Linha 4 do Código 37, é definida através da Ontologia de Localização, apresentada no Código 40 da Seção 5.1.4.3, correlacionada à classe "*Resource*" da Ontologia de Recursos. De forma análoga a propriedade "*closeTo*" é configurada na Linha 5 do Código 37. Vale observar que esta propriedade não é utilizada diretamente na Ontologia de Recursos, ela está correlacionada à propriedade "*locatedIn*" dentro da Ontologia de Localização.

A partir da definição da Ontologia Cliente, neste caso a Ontologia de Localização, e da configuração do OCDRF, a aplicação passa a ser capaz de responder consultas como a apresentada no Código 38.

```
1 <DiscoveryQuery>
2     <CapacityConstraint componentType="Actuator" capacityName="AudioOutput">
3     </CapacityConstraint>

4     <CapacityConstraint componentType="Sensor" capacityName="ActiveDevice">
5         <Attribute name="powerOn" operator="==" value="true" />
6     </CapacityConstraint>

7     <ClientConstraint element="Resource" property="locatedIn" operation="closeTo"
              object="LivingRoom" />

8 </DiscoveryQuery>
```

Código 38 - Exemplo de consulta ao SD utilizando múltiplas restrições

Como descrito anteriormente, o SD suporta consultas com múltiplas restrições. Estas são distintas em dois tipos: (i) restrições sobre as propriedades descritas na Ontologia de Recursos e (ii) restrições sobre as propriedades descritas em Ontologias Cliente.

As Linhas 2-3 e 4-6 do Código 38 apresentam as restrições do tipo um, "*CapacityConstraint*", atribuídas ao recurso buscado. Estas restrições limitam o tipo do componente, "*Actuator*" ou "*Sensor*", e sua capacidade. Esta consulta busca por um recurso que possua como componente um atuador com capacidade de enviar uma mensagem de áudio

(Linha 2), sem requerer nenhum valor especial para seus atributos e um sensor com a capacidade de informar que o equipamento está ou não ligado (Linha 4) com o atributo "*powerOn*" contendo o valor "*true*" (Linha 5).

A Linha 7 apresenta ainda mais uma restrição contida nesta consulta. Nela é utilizada a extensão disponibilizada pelo OCDRF para que a aplicação crie e utilize seus próprios termos para o mapeamento de seu domínio. Nesta consulta a aplicação busca por uma propriedade associada ao nível mais alto de mapeamento na Ontologia de Recursos, isto é, o próprio "*Resource*" (atributo *element*).

A propriedade buscada na Linha 7 é a denominada "*locatedIn*". De posse do valor desta propriedade relacionada à classe *Resource*, através de busca na Ontologia de Recursos, o Serviço de Descoberta é capaz de verificar então se este valor está associado à instância da Ontologia de Localização, com identificação "*LivingRoom*" através da propriedade "*closeTo*" da mesma ontologia. O exemplo no Código 38 considera que o atributo da consulta referente à localização do paciente, "*LivingRoom*", foi obtido anteriormente pela aplicação através de outra consulta ao Serviço de Contexto referente a um recurso capaz de prover a localização do mesmo.

As restrições da consulta realizada ao SD são cumulativas, então a partir da consulta apresentada no Código 38 é esperado que o OCDRF retorne os recursos registrados que possuam a capacidade de enviar mensagens de som, que estejam ligados e que estejam localizados (*located in*) próximos ao (*close to*) cômodo denominado "*LivingRoom*" (sala de estar), mapeado através da Ontologia de Localização. Este tipo de consulta apresenta o grande diferencial da utilização de ontologias para o mapeamento das relações entre os elementos de um domínio.

Vale observar também que além das correlações alcançadas com a utilização de Ontologias Cliente, os termos utilizados na consulta não precisam ser exatamente iguais aos termos utilizados no processo de registro dos recursos envolvidos, dado que a análise semântica dos termos é efetuada com o apoio das ontologias criadas. A aplicação poderia utilizar outros termos para o nome da capacidade, nome do atributo e o objeto da restrição na Ontologia Cliente, por exemplo, como "*SaidaDeAudio*", "*ligado*" e "*SalaDeEstar*", respectivamente, caso estes termos estivessem semanticamente relacionados aos termos registrados originalmente nas ontologias. Esta compatibilidade semântica pode ser definida através do atributo *sameAs* especificado na linguagem OWL.

A flexibilidade semântica oferecida pelo OCDRF para as consultas não se restringe à utilização de sinônimos ou traduções de termos. Também serão localizados recursos que estejam correlacionados através dos outros relacionamentos específicos da ontologia. Por exemplo, caso um recurso esteja localizado em um "*Lavabo*", que esteja localizado na "*SalaDeEstar*"; através da propriedade de transitividade da relação estabelecida por "*locatedIn*", este recurso também será localizado.

5.1.3.7 Implantação da Aplicação

Quando um paciente recebe autorização médica para prosseguir com seu tratamento em sua residência sob a monitoração da aplicação que utiliza o OCDRF como infraestrutura de suporte, uma seqüência de passos deve ser seguida desde o registro das recomendações médicas através de um Plano de Cuidados, passando pela alocação dos equipamentos necessários, até a implantação de sensores e a configuração da infraestrutura, para seu acompanhamento remoto.

Um profissional da área de saúde deve selecionar os equipamentos que devem ser disponibilizados para este paciente e configurar o ambiente de execução da aplicação de telemonitoramento.

Para a instalação da aplicação de telemonitoramento no domicílio do paciente são necessários que os seguintes passos sejam efetuados por parte dos administradores do sistema e dos profissionais de saúde:

1) Criação e configuração do Plano de Cuidados do paciente;
2) Seleção dos equipamentos médicos necessários para a medição de dados fisiológicos;
3) Cadastro de todas as Cadeias de Acionamentos adequadas para a condição de saúde do paciente;
4) Cadastro das informações sobre os destinos das mensagens, como *email* e telefones de familiares e médicos envolvidos nas Cadeias de Acionamentos;
5) Criação dos Agentes de Recursos para os dispositivos disponibilizados, seguindo os procedimentos descritos em (RODRIGUES, 2009a);
6) Configuração das ontologias que descrevem cada recurso a ser utilizado.

Mais alguns passos devem ser efetuados na residência do paciente para a utilização da aplicação:

1) Instalação da aplicação e dos serviços da infraestrutura no computador ou computadores devidos;
2) Configuração da comunicação com a Internet e com o sistema central de monitoramento;
3) Configuração dos serviços de suporte (Registro e Diretório, Descoberta e Contexto);
4) Instalação dos sensores de ambiente;
5) Treinamento do equipamento a ser utilizado para a localização do paciente;
6) Mapeamento da residência do paciente na Ontologia de Localização;
7) Mapeamento dos dispositivos existentes na residência do paciente que podem ser utilizados como meio de *output* para o sistema e configuração de suas descrições em OWL, seguindo padrões pré-definidos para dispositivos de *output* e estabelecendo a conexão com a Ontologia de Localização;
8) Configuração dos ARs para os dispositivos de *output* disponíveis, seguindo os procedimentos descritos em (RODRIGUES, 2009a);
9) Configuração do sistema operacional para que os elementos da arquitetura e todos os ARs para que sejam inicializados logo após a inicialização do mesmo;
10) Ativação de todos os serviços da infraestrutura, de todos os ARs e da aplicação.

Alguns destes passos, relacionados ao OCDRF, são discutidos nas próximas seções detalhando os aspectos relevantes ao uso de ontologias.

5.1.4 Protótipo da Aplicação

Para demonstrar a utilização do OCDRF foi criado um protótipo para o módulo da aplicação executada na residência do paciente, cuja tela principal é apresentada na Figura 19. A imagem principal (A) da Figura 19 representa a residência de um paciente designado a efetuar seu tratamento em seu domicílio e ser acompanhado através de uma aplicação para a

garantia do cumprimento dos passos necessários para a manutenção e melhoria de seu estado de saúde.

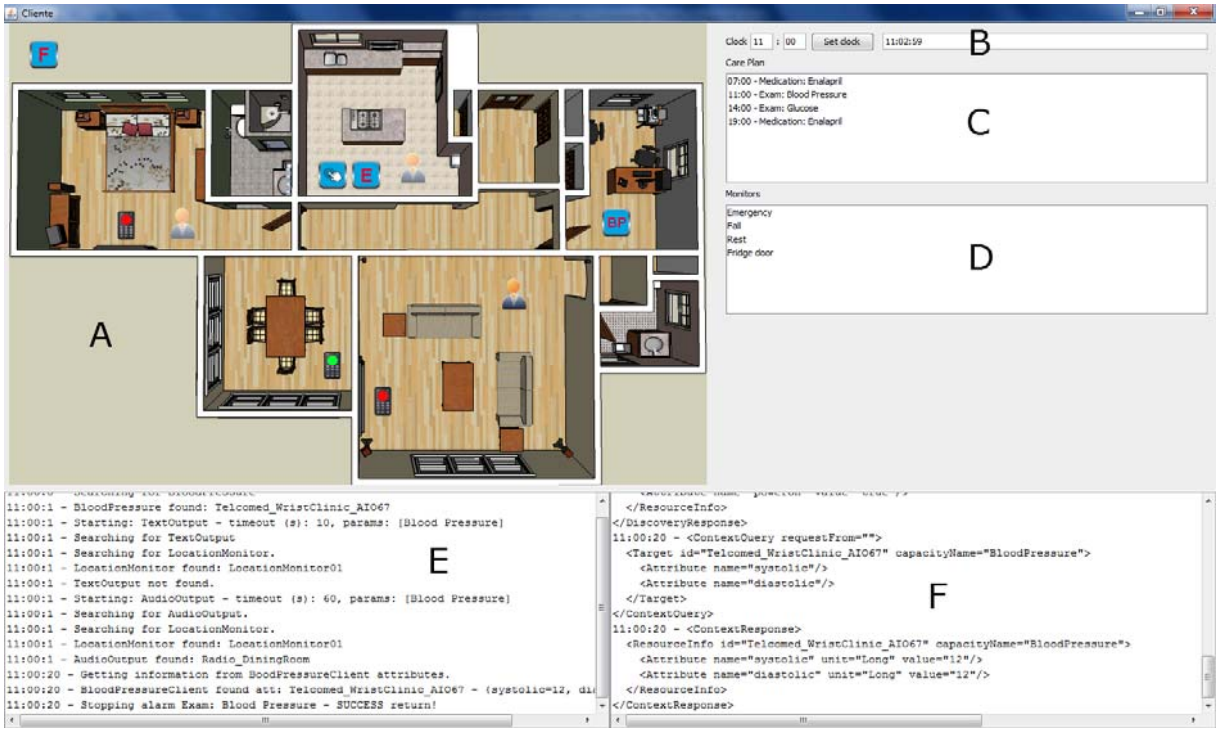

Figura 19 - Foto da tela principal do protótipo da aplicação de telemedicina

No lado superior direito (B) da Figura 19 pode-se ver um relógio que representa a hora atual do sistema e campos à sua esquerda onde é possível alterá-la diretamente para uma hora e minuto específicos. Logo abaixo, ainda à direita (C) na Figura 19 está representada a listagem dos passos cadastrados no Plano de Cuidados, configurado conforme o Código 41, cujas ações serão acionadas de acordo com o horário do sistema. Mais abaixo na direita (D) estão listados os Monitores de Emergência que a aplicação oferece. Na parte inferior da tela (E e F) são apresentadas as mensagens da aplicação. À esquerda (E) pode-se ver o histórico das atividades do protótipo e à direita (F), o histórico das mensagens XML enviadas e recebidas dos serviços providos pelo OCDRF.

Alguns botões contendo letras e outros símbolos estão também presentes no protótipo criado e são detalhados na Figura 20.

F	Imagem que representa um dispositivo capaz de detectar quedas do paciente.

	Imagem que representa um dispositivo capaz de registrar o retorno do paciente para os avisos e alarmes gerados pela aplicação – *Feedback button*.
	Imagem que representa um dispositivo capaz de acionar o estado de emergência.
	Imagem que representa um dispositivo capaz de medir a pressão arterial.
	Imagem que representa a localização do paciente dentro da residência.
	Imagem que representa o controle remoto dos equipamentos de TV e rádio.

Figura 20 - Símbolos presentes na tela do protótipo

5.1.4.1 Dispositivos Utilizados

A Figura 21 apresenta os recursos utilizados no protótipo da aplicação. Estes recursos têm o suporte de Agentes de Recurso e de ontologias criadas para sua representação.

1) Duas TVs da residência, uma na “Sala” e outra no “Quarto1”.
2) Um rádio na Sala de Jantar.
3) Um dispositivo com função de *Feedback button*.
4) Um dispositivo para acionamento de emergências.
5) Um equipamento capaz de detectar quedas.
6) Um sistema de localização em tempo real para localizar o paciente dentro da residência.
7) Um equipamento capaz de realizar medidas fisiológicas do paciente.

Figura 21 - Recursos utilizados no protótipo

5.1.4.2 Ontologia de Recursos

Todos os recursos disponíveis no ambiente para utilização por parte da aplicação cliente devem ser mapeados e cadastrados no sistema. Adicionalmente, Agentes de Recurso

devem ser criados para possibilitar o acesso às operações que tal recurso provê, como por exemplo a monitoração de seu estado (ligado ou desligado).

Para cada recurso apresentado na Figura 21 foi necessária então sua descrição de acordo com a Ontologia de Recursos utilizada no OCDRF e o desenvolvimento de um Agente de Recurso correspondente.

Dentre os equipamentos que foram registrados e utilizados no protótipo, será detalhada nesta seção a descrição do equipamento de TV, como exemplo.

Para o envio de mensagens de aviso para o paciente é necessário que estejam disponíveis no ambiente domiciliar equipamentos capazes de receber e exibir mensagens. Tais equipamentos precisam então ser descritos através da Ontologia de Recursos e serem registrados junto ao SRD. Além disso, dado que a aplicação de telemedicina se utiliza da localização destes equipamentos para o envio mais adequado dos avisos para o paciente, as descrições de tais recursos devem possuir a informação do cômodo da residência onde eles estão instalados.

Através do registro de recursos com capacidade de envio de avisos, da classificação dos cômodos da moradia e da informação de localização do paciente é possível determinar a proximidade do paciente em relação aos dispositivos que estão disponíveis. De posse da informação de que o paciente se encontra na sala e que o televisor da sala está ligado, por exemplo, a aplicação pode utilizar este meio para o envio de uma mensagem de vídeo, áudio ou texto ao paciente.

Na sala representada na Figura 19, está instalada uma TV conectada a um equipamento conversor (*set-top box*), como o VT7200E da Visiontec (VISIONTEC, 2011) e é apresentado seu controle remoto. Para que esta TV seja localizada pelo sistema, é necessário que suas características sejam registradas no OCDRF. Isto é feito através do registro de sua descrição em formato OWL como apresentado no Código 39.

```
1 <?xml version="1.0"?>
2 <!DOCTYPE owl [
3  <!ENTITY res "http://localhost/ocdrf/conf/Resource.owl#" >
4  <!ENTITY loc "http://localhost/ocdrf/conf/Location.owl#" >
5  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
6  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
7 ]>

8 <rdf:RDF
9  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
10  xmlns:rdfs= "http://www.w3.org/2000/01/rdf-schema#"
11  xmlns:owl = "&owl;"
12  xmlns:xsd = "&xsd;"
13  xmlns:loc = "&loc;"
14  xml:base  = "&res;"
15  xmlns     = "&res;">

16 <owl:Ontology rdf:about="">
17 <rdfs:comment>LivingRoom TV Description</rdfs:comment>
18 </owl:Ontology>

19 <Resource rdf:ID="TV_LivingRoom">
20  <hasName        rdf:datatype="&xsd;string">LivingRoom TV</hasName>
21  <hasComponent   rdf:resource="#TV_LivingRoom_Active_Engine" />
22  <hasComponent   rdf:resource="#TV_LivingRoom_Write_Engine" />
23  <hasVendor      rdf:datatype="&xsd;string">Visiontec</hasVendor>
24  <hasModel       rdf:datatype="&xsd;string">VT7200E</hasModel>
25  <invokedBy      rdf:resource="#TV_LivingRoom_WS" />
26  <loc:locatedIn  rdf:resource="&loc;LivingRoom" />
27 </Resource>

28 <Sensor rdf:ID="TV_LivingRoom_Active_Engine">
29  <hasName            rdf:datatype="&xsd;string">Active_Engine</hasName>
30  <hasCapacity        rdf:resource="#ActiveDevice" />
31  <hasInvokeOperation rdf:resource="#TV_LivingRoom_Active_Engine_Op" />
32 </Sensor>

33 <InvokeMethod rdf:ID="TV_LivingRoom_WS">
34  <invokeString        rdf:datatype="&xsd;string">
        http://localhost:1988/TVAgent_LivingRoom?wsdl</invokeString>
35  <hasTechnology       rdf:resource="#webservice" />
36 </InvokeMethod>

37 <InvokeOperation rdf:ID="TV_LivingRoom_Active_Engine_Op">
38  <operationName     rdf:datatype="&xsd;string">
       getTVLivinRoom_Active_EngineState</operationName>
39  <invokeReturnType  rdf:datatype="&xsd;string">String</invokeReturnType>
40 </InvokeOperation>

41 <InputCapacity rdf:ID="ActiveDevice">
42  <has Attribute     rdf:resource="#powerOn" />
43 </InputCapacity>

44 <Attribute rdf:ID="powerOn">
45  <hasName           rdf:datatype="&xsd;string">powerOn</hasName>
46  <hasAttributeType  rdf:resource="#Dynamic" />
47  <hasAttributeUnit  rdf:resource="#Boolean" />
48 </Attribute>

49 <Actuator rdf:ID="TV_LivingRoom_Write_Engine">
50  <hasCapacity        rdf:resource="#TextOutput" />
51  <hasInvokeOperation rdf:resource="#TV_LivingRoom_Write_Engine_Op" />
52 </Actuator>

53 <InvokeOperation rdf:ID="TV_LivingRoom_Write_Engine_Op">
54  <operationName  rdf:datatype="&xsd;string">writeMessage</operationName>
```

```
55  <invokeReturnType  rdf:datatype="&xsd;string">String</invokeReturnType>
56 </InvokeOperation>

57 <owl:Thing rdf:ID="EquipamentoAtivo">
58  <owl:sameAs     rdf:resource="#ActiveDevice"/>
59 </owl:Thing>

60 <owl:Thing rdf:ID="ligado">
61  <owl:sameAs     rdf:resource="#powerOn"/>
62 </owl:Thing>

...

63 </rdf:RDF>
```

Código 39 - Características do equipamento televisor disponível na sala de estar

A definição do recurso apresentada no Código 39 será utilizada pelo protótipo da aplicação cliente de telemedicina. A Linha 3 do Código 39 faz referência à Ontologia de Recursos (apresentada na Figura 4) e a Linha 4 faz referência à Ontologia de Localização, Código 40, a ser descrita na Seção 5.1.4.3.

As Linhas 8-15 indicam precisamente os vocabulários que estão sendo usados nesta ontologia, através da declaração do conjunto de *namespaces* XML (W3C4, 2009). A Linha 13 identifica o *namespace* da Ontologia de Localização e a Linha 15 define o *namespace* padrão, que determina que os termos não prefixados se referem à Ontologia de Recursos.

As Linhas 16-18 apresentam o cabeçalho da ontologia. As Linhas 19-27 representam uma instância de *Resource*. A Linha 20 apresenta uma descrição do recurso. As propriedades *hasVendor* (Linha 23) e *hasModel* (Linha 24) também são apenas descritivas e apresentam o fabricante e o modelo do equipamento.

Neste televisor foram classificados dois componentes, que estão configurados nas Linhas 21 e 22 através da propriedade *hasComponent*. Nestas linhas são apresentadas as referências para os componentes descritos pelas Linhas 28-32 e 49-52. Conforme apresentado na Ontologia de Recursos, um componente pode ser um *Sensor* ou um *Actuator*.

O sensor de ID "*TV_LivingRoom_Active_Engine*" (Linha 28) possui a capacidade de determinar se o equipamento está ou não ligado, através da propriedade *hasCapacity* (Linha 30) associada à *Capacity* "*ActiveDevice*" descrita nas Linhas 41-43. Esta capacidade possui um atributo denominado "*powerOn*" (Linha 42).

O atributo *powerOn*, descrito nas linhas 44-48, é um atributo dinâmico (Linha 46), isto é, seu valor deve ser definido em tempo de execução e resgatado através de consultas ao Serviço de Contexto, e representa uma informação *booleana*, com valores possíveis como *true* ou *false*.

Este recurso possui associado a ele um Agente de Recurso que pode ser invocado através da interface definida pela *string* de conexão representada pela propriedade *invokedBy* (Linha 25), cuja definição (Linhas 33-36) inclui a *invokeString*, neste caso uma URL por se tratar de um serviço web.

Para que o Serviço de Contexto possa acessar um sensor, como o "*TV_LivingRoom_Active_Engine*" definido na Linha 28, para recuperar os valores de atributos dinâmicos, ele utiliza a URL definida por *invokeString* e a operação que implementa este sensor, como definido pela propriedade *hasInvokeOperation* (Linha 31). A definição desta propriedade para este sensor (Linhas 37-40) inclui seu nome, *operationName* (Linha 38), e o tipo de dado que esta operação irá retornar *invokeReturnType* (Linha 39).

O atuador que compõe este recurso está descrito nas Linhas 49-52 e possui associado a ele a capacidade de exibir mensagens de texto (Linha 50). A operação que implementa este atuador está representada nas Linhas 53-56.

As Linhas 57-59 e 60-62 apresentam os termos "*EquipamentoAtivo*" e "ligado" sendo configurados para terem o mesmo significado semântico dos termos "*ActiveDevice*"e "*powerOn*". Desta forma os termos "*EquipamentoAtivo*" e "*ActiveDevice*" podem ser usados de forma intercambiável, assim como os termos "ligado" e "*powerOn*".

5.1.4.3 Ontologia de Localização

O protótipo da aplicação de telemedicina se utiliza de uma ontologia, denominada neste trabalho como Ontologia de Localização, para realizar o mapeamento dos cômodos existentes na residência do paciente. Tal ontologia descreve a correlação entre os cômodos, incluindo sua composição e a proximidade entre os mesmos.

Esta ontologia pode ser agregada ao OCDRF, de forma que as consultas efetuadas a esta infraestrutura possam se utilizar dos termos definidos na mesma. Este ponto de extensão fornecido pela infraestrutura para a utilização de ontologias providas pela aplicação cliente é detalhado na Seção 3.2.

O mapeamento dos cômodos da residência realizado através da Ontologia de Localização (Código 40) é utilizado então pela aplicação para obter as informações sobre os recursos mais próximos do paciente. Através do mapeamento da residência será possível à aplicação saber que o “Quarto2”, está próximo da “Sala” o suficiente para que um aviso sonoro emitido na sala seja ouvido no “Quarto2”. Este mapeamento é bastante útil em residências extensas ou que possuam mais de um pavimento e para pacientes com problemas de audição.

A descrição do equipamento de TV da sala do paciente no protótipo (Linha 26 do Código 39) informa que este recurso está localizado no ambiente denominado “*LivingRoom*”. O termo “*loc*”, utilizado em 2 pontos nesta linha, faz referência a Ontologia de Localização, apresentada no Código 40.

```
 1 <?xml version="1.0"?>
 2 <!DOCTYPE owl [
 3   <!ENTITY loc "http://localhost/ocdrf/conf/Location.owl#" >
 …
 4 <owl:Ontology rdf:about="">
 5    <rdfs:label>Location Ontology - Client Ontology</rdfs:label>
 6 </owl:Ontology>

 7 <owl:Class rdf:ID="Location" />

 8 <owl:ObjectProperty rdf:ID="locatedIn">
 9    <rdf:type rdf:resource="&owl;TransitiveProperty" />
10    <rdfs:domain rdf:resource="&owl;Thing" />
11    <rdfs:range rdf:resource="#Location" />
12 </owl:ObjectProperty>

13 <owl:ObjectProperty rdf:ID="closeTo">
14    <rdf:type rdf:resource="&owl;SimetricProperty" />
15    <rdfs:domain rdf:resource="#Location" />
16    <rdfs:range rdf:resource="#Location" />
17 </owl:ObjectProperty>

18 <Location rdf:ID="Home" />

19 <Location rdf:ID="Bedroom1">
20    <locatedIn rdf:resource="#Home" />
21 </Location>

22 <Location rdf:ID="Bedroom2">
23    <locatedIn rdf:resource="#Home" />
24 </Location>

25 <Location rdf:ID="LivingRoom">
26    <locatedIn rdf:resource="#Home" />
27    <closeTo rdf:resource="#Bedroom2" />
28 </Location>
```

```
29 <Location rdf:ID="DiningRoom">
30    <locatedIn rdf:resource="#Home" />
31    <closeTo rdf:resource="#LivingRoom" />
32 </Location>

33 <Location rdf:ID="Bathroom1">
34    <locatedIn rdf:resource="#Bedroom1" />
35 </Location>

36 <Location rdf:ID="Bathroom2">
37    <locatedIn rdf:resource="#Home" />
38    <closeTo rdf:resource="#Bedroom2" />
39 </Location>

40 <Location rdf:ID="Kitchen">
41    <locatedIn rdf:resource="#Home" />
42 </Location>

43 </rdf:RDF>
```

Código 40 - Ontologia de Localização mapeando a residência de um paciente

No Código 40 são definidas uma classe, denominada "*Location*" (Linha 7), e duas propriedades, denominadas "*locatedIn*" e "*closeTo*" (Linhas 8-12 e 13-17). São descritos também os ambientes encontrados na residência do paciente que serão usados pelo sistema de telemedicina para prover o paciente com o máximo de iteratividade utilizando os equipamentos disponíveis em seu lar.

A classe "*Location*" representa cada cômodo da residência e a residência propriamente dita. Os cômodos são instâncias de *Location* denominados "*Bedroom1*" (Linha 19), "*Bedroom2*" (Linha 22), "*LivingRoom*" (Linha 25), "*DiningRoom*" (Linha 29), "*Bathroom1*" (Linha 33), "*Bathroom2*" (Linha 36) e "*Kitchen*" (Linha 40). Para finalizar, "*Home*" (Linha 18) é a instância que representa a casa em si.

A propriedade "*locatedIn*" (Linhas 8-12) é uma propriedade transitiva (Linha 9) cujo domínio é um elemento de qualquer tipo da linguagem OWL e seu *range*, a classe *Location*. Esta propriedade representa que qualquer componente pode estar localizado em uma instância de localidade. A transitividade permite inferir que se um elemento A está localizado em um cômodo B e este cômodo está localizado em um cômodo maior C, então, o elemento A está também localizado no cômodo C.

A propriedade "*closeTo*" (Linhas 13-17) é uma propriedade simétrica (Linha 14) cujo domínio e *range* são uma instância de *Location*. Esta propriedade representa a proximidade

entre cômodos. A simetria garante que se um cômodo A está próximo de B, o contrário também é verdadeiro, então, B está próximo de A.

As propriedades "*locatedIn*" e "*closeTo*" são utilizadas na aplicação de telemedicina para a descrição da interligação entre os cômodos da residência e para a localização de equipamentos e do paciente no interior da casa. As Linhas 29-32 descrevem por exemplo, que a sala de jantar "*DiningRoom*" está localizada na casa do paciente "*Home*" e que fica próxima à sala de estar "*LivingRoom*".

5.1.4.4 Planos de Cuidados

O Código 41 apresenta o Plano de Cuidados configurado para o protótipo da aplicação cliente.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <CarePlan>
 3   <CarePlanItem>
 4         <StartTime>07:00</StartTime>
 5         <CarePlanItemTypeId>1</CarePlanItemTypeId>
 6         <Description>Enalapril</Description>
 7         <FeedbackInput>FeedBackButton</FeedbackInput>
 8   </CarePlanItem>
 9   <CarePlanItem>
10        <StartTime>11:00</StartTime>
11        <CarePlanItemTypeId>3</CarePlanItemTypeId>
12        <Description>Blood Pressure</Description>
13        <FeedbackInput>BloodPressure</FeedbackInput>
14   </CarePlanItem>
15   <CarePlanItem>
16        <StartTime>14:00</StartTime>
17        <CarePlanItemTypeId>3</CarePlanItemTypeId>
18        <Description>Glucose</Description>
19        <FeedbackInput>GlucoseMonitor</FeedbackInput>
20   </CarePlanItem>
21   <CarePlanItem>
22        <StartTime>19:00</StartTime>
23        <CarePlanItemTypeId>1</CarePlanItemTypeId>
24        <Description>Enalapril</Description>
25        <FeedbackInput>FeedBackButton</FeedbackInput>
26   </CarePlanItem>
27 </CarePlan>
```

Código 41 - Exemplo de Plano de Cuidados

Um exemplo de um item no Plano de Cuidados disponibilizado no protótipo foi configurado nas Linhas 3-8 do Código 41. Às 7h da manhã (Linha 4) a aplicação deve disparar uma mensagem de tipo 1 (Linha 5) informando sobre a necessidade de se ministrar o medicamento “Enalapril” (Linha 6) e que o dispositivo de tipo “*FeedBackButton*” (Linha 7) deve ser monitorado para que seja dado como concluído este passo do Plano de Cuidados. O Plano de Cuidados deve ser definido conforme o XSD apresentado no Código 42.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       attributeFormDefault="unqualified" elementFormDefault="qualified">
 3 <xsd:element name="CarePlan">
 4   <xsd:complexType>
 5   <xsd:sequence>
 6     <xsd:element name="CarePlanItem" maxOccurs="unbounded">
 7       <xsd:complexType>
 8         <xsd:sequence>
 9           <xsd:element name="StartTime" type="xsd:time" />
10           <xsd:element name="CarePlanItemTypeId" type="xsd:positiveInteger" />
11           <xsd:element name="Description" type="xsd:string" />
12           <xsd:element name="FeedbackInput" type="xsd:string" />
13         </xsd:sequence>
14       </xsd:complexType>
15     </xsd:element>
16   </xsd:sequence>
17   </xsd:complexType>
18 </xsd:element>
19 </xsd:schema>
```

Código 42 - XSD para descrição do Plano de Cuidados

A Linha 6 do Código 42 define que um Plano de Cuidados pode possuir múltiplos elementos (*CarePlanItem*). Estes elementos possuem associados a eles os atributos obrigatórios e de cardinalidade máxima um (*default*), “*StartTime*” do tipo horário (Linha 9), “*CarePlanItemTypeId*” do tipo inteiro positivo (Linha 10) e “*Description*” do tipo texto (Linha 11). Além destes atributos, *CarePlanItem* também possui o atributo opcional e de cardinalidade máxima um, “*FeedbackInput*” do tipo texto (Linha 12).

StartTime (Linha 9) estabelece o horário para o disparo do passo do Plano de Cuidados.

CarePlanItemTypeId (Linha 10) é uma referência ao tipo de item que pode existir no Plano de Cuidados, cuja definição deve estar em conformidade com a especificação apresentada no Código 44.

Description (Linha 11) é uma descrição do passo a ser efetuado. Esta descrição será utilizada na montagem dos avisos a serem enviados para o paciente ou responsável cadastrado.

FeedbackInput (Linha 12) representa o tipo de dispositivo que deve ser monitorado para que a aplicação obtenha a confirmação sobre a execução de um item no Plano de Cuidados. *FeedbackInput* deve possuir como valor o nome da capacidade que um dispositivo deve possuir para que a alteração de seu contexto sirva como confirmação de execução da atividade estabelecida no Plano de Cuidados. Caso o item seja apenas uma mensagem, que não precise de nenhuma entrada ou confirmação por parte dos envolvidos, esta informação não deve ser preenchida.

Cada item no Plano de Cuidados deve estar associado a um tipo. Isto é realizado através do "*CarePlanItemTypeId*" representado nas Linhas 5, 11, 17 e 23 do Código 41. O Código 43 apresenta os tipos de itens para o Plano de Cuidados disponíveis para serem usados pelo protótipo da aplicação desenvolvido.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <CarePlanItemTypes>
 3   <ItemType>
 4         <Id>1</Id>
 5         <Description>Medication Type 1</Description>
 6         <MessageChainId>1</MessageChainId>
 7    </ItemType>
 8    <ItemType>
 9         <Id>2</Id>
10         <Description>Medication Type 2</Description>
11         <MessageChainId>2</MessageChainId>
12    </ItemType>
13    <ItemType>
14         <Id>3</Id>
15         <Description>Exam</Description>
16         <MessageChainId>2</MessageChainId>
17    </ItemType>
18 </CarePlanItemTypes>
```

Código 43 - Exemplo de Tipos de Itens no Plano de Cuidados

Como exemplo de tipos de item no Plano de Cuidados disponibilizados no protótipo, as Linhas 3-7 do Código 43 representam que o item de identificação 1 (Linha 4), descrito como "*Medication Type 1*" (Linha 5) está associado à Cadeia de Acionamentos identificada pelo número 1 (Linha 6).

O cadastro de tipos de item no Plano de Cuidados deve seguir a especificação apresentada no XSD do Código 44.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       attributeFormDefault="unqualified" elementFormDefault="qualified">
 3 <xsd:element name="CarePlanItemTypes">
 4  <xsd:complexType>
 5  <xsd:sequence>
 6    <xsd:element name="ItemType" minOccurs="0" maxOccurs="unbounded">
 7      <xsd:complexType>
 8        <xsd:sequence>
 9          <xsd:element name="Id" type="xsd:string" />
10          <xsd:element name="Description" type="xsd:string" />
11          <xsd:element name="MessageChainId" type="xsd:positiveInteger" minOccurs="0" />
12        </xsd:sequence>
13      </xsd:complexType>
14    </xsd:element>
15   </xsd:sequence>
16   </xsd:complexType>
17 </xsd:element>
18 </xsd:schema>
```

Código 44 - XSD para descrição de Tipos de Itens do Plano de Cuidados

A Linha 6 do Código 44 define que podem ser cadastrados múltiplos tipos de itens no Plano de Cuidados (*ItemType*). Cada tipo possui associado a ele os atributos obrigatórios e de cardinalidade máxima (*default*) um, "*Id*" (Linha 9), identificador do item, e "*Description*" (Linha 10), descrição para leitura humana, ambos do tipo texto. Além destes atributos, um tipo de item no Plano de Cuidados também possui o atributo opcional e de cardinalidade máxima (*default*) um, "*MessageChainId*" do tipo inteiro positivo (Linha 11). *MessageChainId* é uma referência às Cadeias de Acionamentos. Cada item no Plano de Cuidados que precise de intervenção por parte do paciente, profissionais de saúde ou familiares, terá associado a ele uma Cadeia de Acionamentos.

Vários tipos de item podem estar associados a uma mesma cadeia de acionamentos, como pode ser visto nas Linhas 11 e 16 do Código 43, onde dois tipos de item estão associados à mesma cadeia.

5.1.4.5 Cadeias de Acionamento

O conjunto de Cadeias de Acionamentos utilizado no protótipo é apresentado no Código 45.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <MessageChainSet>
 3    <MessageChain>
 4         <Id>1</Id>
 5         <Targets>
 6             <Target>
 7                <Priority>1</Priority>
 8                <DestinationId>1</DestinationId>
 9                <Timeout>5</Timeout>
10             </Target>
11             <Target>
12                <Priority>2</Priority>
13                <DestinationId>4</DestinationId>
14                <Timeout>5</Timeout>
15            </Target>
16            <Target>
17                <Priority>3</Priority>
18                <DestinationId>3</DestinationId>
19                <Timeout>5</Timeout>
20            </Target>
21            <Target>
22                <Priority>4</Priority>
23                <DestinationId>2</DestinationId>
24                <Timeout>5</Timeout>
25            </Target>
26        </Targets>
27   </MessageChain>
28   <MessageChain>
29        <Id>2</Id>

          . . .
30 </MessageChainSet>
```

Código 45 - Exemplo de conjunto de Cadeias de Acionamentos

Um exemplo de Cadeia de Acionamentos disponibilizado no protótipo foi configurado nas Linhas 3-27 do Código 45. A cadeia de identificação 1 (Linha 4) possui os destinos configurados entre as Linhas 5-26. As Linhas 6-10 apresentam o primeiro alvo da cadeia, de prioridade 1 (Linha 7), associado ao destino de identificação igual a 1 (Linha 8), configurado com *timeout* igual a 5 minutos (Linha 9). Todos os demais alvos das mensagens também estão configurados com *timeout* igual a 5 minutos.

As cadeias de Acionamentos devem ser definidas conforme o XSD apresentado no Código 46.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       attributeFormDefault="unqualified" elementFormDefault="qualified">
 3 <xsd:element name="MessageChainSet">
 4   <xsd:complexType>
 5   <xsd:sequence>
 6     <xsd:element name="MessageChain" minOccurs="0" maxOccurs="unbounded">
 7       <xsd:complexType>
 8         <xsd:sequence>
 9           <xsd:element name="Id" type="xsd:positiveInteger" minOccurs="0" />
10           <xsd:element name="Targets" minOccurs="0">
11              <xsd:complexType>
12                <xsd:sequence>
13                  <xsd:element name="Target" minOccurs="1" maxOccurs="unbounded">
14                    <xsd:complexType>
15                       <xsd:sequence>
16                         <xsd:element name="Priority" type="xsd:positiveInteger" />
17                         <xsd:element name="DestinationId" type="xsd:positiveInteger" />
18                         <xsd:element name="Timeout" type="xsd:positiveInteger" />
19                       </xsd:sequence>
20                    </xsd:complexType>
21                  </xsd:element>
22                </xsd:sequence>
23              </xsd:complexType>
24           </xsd:element>
25         </xsd:sequence>
26       </xsd:complexType>
27     </xsd:element>
28   </xsd:sequence>
29  </xsd:complexType>
30 </xsd:element>
31 </xsd:schema>
```

Código 46 - XSD para descrição do conjunto de Cadeia de Acionamentos

A Linha 6 do Código 46 define que podem haver múltiplas Cadeias de Acionamentos (*MessageChain*) no sistema. Cada Cadeia de Acionamentos é composta de atributos obrigatórios e de cardinalidade máxima (default) um, "*Id*" (Linha 9), identificador da cadeia, do tipo inteiro positivo e "*Targets*" (Linhas 10-24), alvos para o envio. O atributo "*Targets*" é por sua vez, composto de um a infinitos alvos (Linha 13). Cada "*Target*" é composto dos atributos obrigatórios e de cardinalidade mínima e máxima um, "*Priority*" (Linha 16), prioridade, "*DestinationId*" (Linha 17), identificador do destino e "*Timeout*" (Linha 18), limite de tempo em minutos para o aviso, todos de tipo inteiro positivo.

O atributo “*Id*” é utilizado para identificar a Cadeia de Acionamentos. Conforme os temporizadores cadastrados para um destino de um passo do plano, representados pelo atributo “*Timeout*”, alcancem seu limite, novos envolvidos cadastrados são acionados, de acordo com o atributo “*Priority*”, o que pode incluir familiares, médicos e a central de monitoramento de acordo com a condição de saúde de cada paciente e a urgência de que a ação pendente seja executada. A Cadeia de Acionamentos deve ser configurada previamente na aplicação, o que pode ser realizado local ou remotamente, para cada tipo de ação prevista no Plano de Cuidados.

Cada alvo na Cadeia de Acionamentos precisa estar associado a um destino. As Linhas 8, 13, 18 e 23 do Código 45 possuem a identificação do destino do aviso. O Código 47 apresenta os destinos configurados no protótipo.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <Destinations>
 3   <Destination>
 4      <Id>1</Id>
 5      <Description>TextOutput in the room the patience is</Description>
 6      <Device>TextOutput</Device>
 7      <Location>locatedIn</Location>
 8   </Destination>
 9   <Destination>
10      <Id>2</Id>
11      <Description>SMSOutput to the doctor</Description>
12      <Device>SMSOutput</Device>
13      <PhoneNumber>1212xxxxxxx</PhoneNumber>
14   </Destination>
15   <Destination>
16      <Id>3</Id>
17      <Description>SMSOutput to the daughter</Description>
18      <Device>SMSOutput</Device>
19      <PhoneNumber>1212yyyyyyy</PhoneNumber>
20   </Destination>
21   <Destination>
22      <Id>4</Id>
23      <Description>RadioOutput close to where the patience is</Description>
24      <Device>RadioOutput</Device>
25      <Location>closeTo</Location>
26    </Destination>
27 </Destinations>
```

Código 47 - Exemplo de cadastro de destinos para a Cadeia de Acionamentos

Como exemplo de destino disponibilizado no protótipo, as Linhas 3-8 do Código 47 representam o destino de identificação 1 (Linha 4), descrito na Linha 5. Este está associado ao dispositivo com a capacidade de “*TextOutput*” (Linha 6) que deve estar localizado no mesmo

cômodo onde se encontra (Linha 7) o destino. Nos destinos de identificação 2 e 3 o atributo *PhoneNumber* (número do telefone do destinatário) (Linhas 13 e 19) é utilizado ao invés da localização relativa à residência. Os elementos *Location* e *PhoneNumber* são mutuamente exclusivos.

O Código 48 a seguir apresenta a especificação em XSD que deve ser seguida para o cadastro de destinos apresentado no Código 47.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       attributeFormDefault="unqualified" elementFormDefault="qualified">
 3 <xsd:element name="Destinations">
 4   <xsd:complexType>
 5   <xsd:sequence>
 6     <xsd:element name="Destination" minOccurs="0" maxOccurs="unbounded">
 7       <xsd:complexType>
 8         <xsd:sequence>
 9           <xsd:element name="Id" type="xsd:positiveInteger" />
10           <xsd:element name="Description" type="xsd:string" />
11           <xsd:element name="Device" type="xsd:string" />
12           <xsd:element name="Location" type="xsd:string" minOccurs="0" />
13           <xsd:element name="PhoneNumber" type="xsd:string" minOccurs="0" />
14            </xsd:sequence>
15       </xsd:complexType>
16     </xsd:element>
17   </xsd:sequence>
18  </xsd:complexType>
19 </xsd:element>
20 </xsd:schema>
```

Código 48 - XSD para descrição de Destinos

A Linha 6 do Código 48 define que podem haver múltiplos destinos cadastrados no sistema. Cada destino é composto de atributos obrigatórios e de cardinalidade máxima (*default*) um, "*Id*" do tipo inteiro positivo (Linha 9), "*Description*" do tipo texto (Linha 10) e "*Device*" também do tipo texto (Linha 11), além dos atributos "*Location*" e "*PhoneNumber*", ambos do tipo texto e opcionais.

De forma análoga aos itens no Plano de Cuidados, os Monitores de Emergência do sistema podem ser configurados através de informações no formato XML. Sua configuração não será detalhada nesta dissertação.

5.1.5 Cenário de Utilização

A seguir é descrito um cenário de exemplo de utilização da aplicação.

Um paciente que sofre de hipertensão é acompanhado remotamente através da aplicação de telemedicina proposta.

Quando a aplicação é iniciada, ocorre uma varredura dos recursos registrados no sistema e são realizadas validações sobre a disponibilidade de certos recursos. É verificada, por exemplo, a disponibilidade de um elemento do tipo *FeedBackButton*, essencial para o retorno do paciente nos avisos de medicação enviados pela aplicação e para a estabilização de um estado de emergência. É verificada também a existência de um e apenas um recurso com capacidade do tipo *LocationMonitor* na residência, para que não haja confusão quanto à localização do objeto maior de interação, o próprio paciente.

Durante seu funcionamento, a aplicação verifica a partir do Plano de Cuidados que o paciente precisa ter sua pressão sanguínea medida. A aplicação verifica então a Cadeia de Acionamentos a ser seguida por este passo do Plano de Cuidados. Esta define que o primeiro a ser informado sobre o passo a ser executado é o próprio paciente, através de uma mensagem de texto a ser enviada utilizando o equipamento com esta capacidade, que esteja disponível no mesmo cômodo onde o paciente se encontra.

Para que a aplicação seja capaz de enviar a mensagem correta para o paciente, são necessários então os seguintes passos:

(i) localizar o dispositivo capaz de realizar a medida necessária dentro da residência do paciente;

(ii) consultar, através do recurso de localização, o cômodo onde o paciente se encontra; e

(iii) verificar a disponibilidade de dispositivos registrados que possuam a capacidade necessária para o envio do aviso e estejam localizados onde especificado pela aplicação (neste caso, no cômodo onde se encontra o paciente).

A partir da identificação do recurso necessário para envio do aviso, a aplicação deve:

(iv) enviar o aviso através do recurso encontrado e

(v) iniciar o monitoramento sobre a medição da pressão sanguínea, para dar por concluído este passo no Plano de Cuidados.

A Figura 22 apresenta o diagrama contendo as atividades a serem efetuadas pela aplicação após o disparo de um passo no Plano de Cuidados.

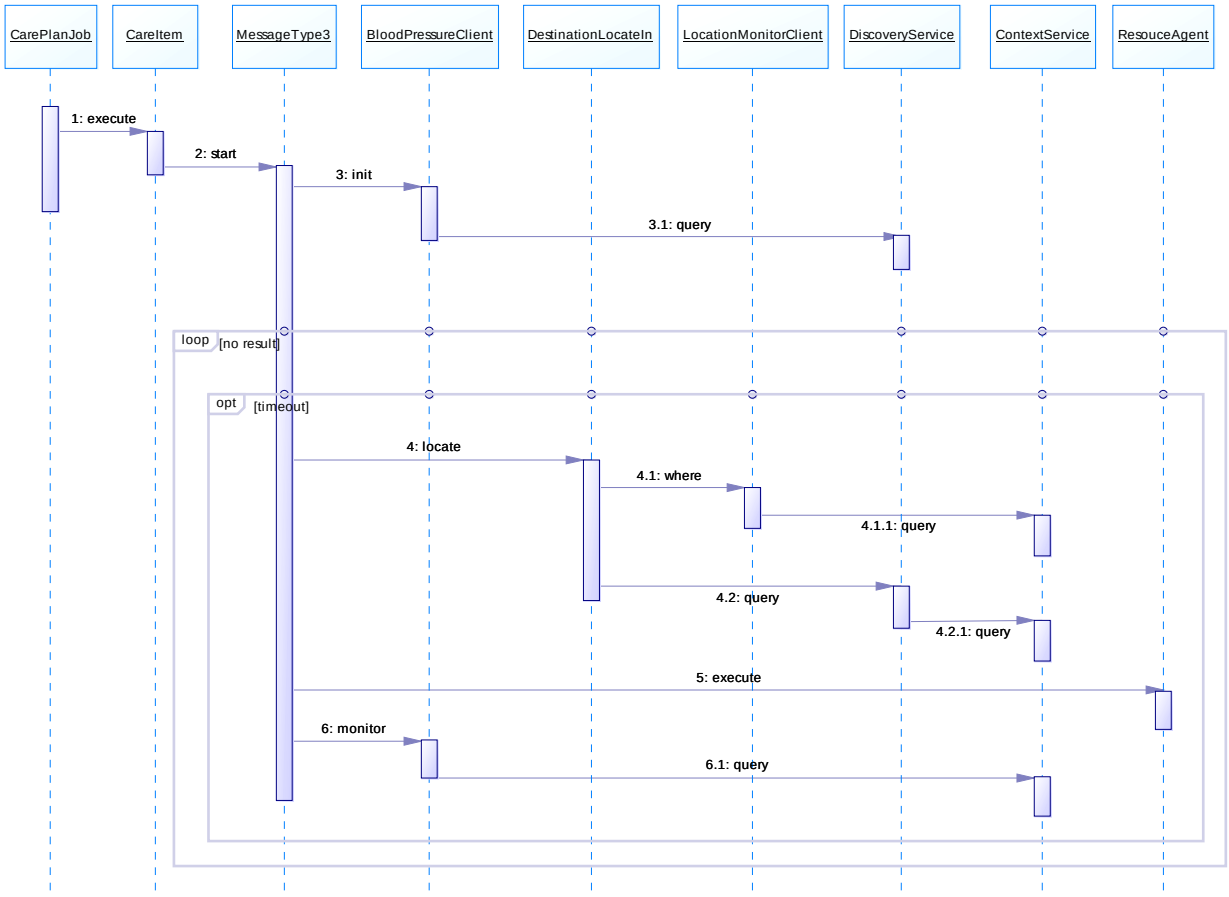

Figura 22 - Atividades executadas pela aplicação depois do disparo do Plano de Cuidados

Como primeiro passo, a aplicação irá procurar por um dispositivo capaz de efetuar a medição de pressão sanguínea, que esteja disponível na moradia do paciente (Interação 3.1 da Figura 22). Para isto o Serviço de Descoberta do OCDRF é acionado com a missão de retornar os dispositivos registrados que possuam esta capacidade. O Código 49 apresenta a consulta que deve ser enviada pela aplicação para o OCDRF.

```
1 <DiscoveryQuery>
2     <CapacityConstraint componentType="Sensor" capacityName="BloodPressure">
3     </CapacityConstraint>
4     <ClientConstraint element="Resource" property="locatedIn"
                operation="locatedIn" object="Home" />
```

```
5 </DiscoveryQuery>
```

Código 49 - Consulta ao SD para obter um equipamento para medição de pressão sanguínea

A consulta apresentada no Código 49 para o Serviço de Descoberta segue a especificação em XSD listada no Código 13. A Linha 2 define que a aplicação precisa localizar um recurso composto de um sensor com a capacidade de medir a pressão sanguínea (*Blood Pressure*) e a Linha 4 define que este recurso deve estar localizado dentro da residência do paciente, "*Home*". Para que a consulta retorne o esperado, um recurso com esta capacidade deve ter se registrado no SRD definindo as suas características de acordo com a Ontologia de Recursos e fazendo referência a um elemento contido na Ontologia de Localização.

O Código 50 apresenta uma possível resposta à consulta efetuada no Código 49.

```
1 <Response>
2       <ResourceInfo id="Telcomed_WristClinic_AI067" componentType="Sensor"
        capacityName="BloodPressure" />
3 </Response>
```

Código 50 - Exemplo de resposta com um equipamento para medição de pressão sanguínea

O Serviço de Descoberta (SD) retorna a identificação do equipamento para medição de pressão sanguínea localizado, como apresentado na Linha 2 do Código 50. É importante ressaltar que através da análise da Ontologia de Localização, definida no Código 40 e configurada como no Código 37, o SD é capaz de localizar qualquer recurso que esteja localizado dentro da residência, uma vez que a propriedade "*locatedIn*" é transitiva, e que os recursos estejam registrados como localizados em um dos cômodos da residência.

Em seguida, a aplicação busca a localização do paciente (Interação 4.1.1 da Figura 22). Isto é realizado através de uma consulta ao Serviço de Contexto informando o identificador do equipamento utilizado para a monitoração da localização do paciente na residência. O identificador deste equipamento foi obtido pela aplicação durante sua inicialização através de uma consulta ao SD.

De posse da localização do paciente, a aplicação procura pelo dispositivo adequado para enviar para o paciente o aviso de solicitação para a execução do passo pendente do Plano de Cuidados (Interação 4.2 da Figura 22). O aviso para medição de pressão sanguínea, neste caso, será primeiramente enviado por texto para o paciente através de um equipamento

localizado no mesmo cômodo onde ele se encontra. Para isto deve ser enviado para o Serviço de Descoberta uma consulta buscando por um equipamento que esteja ligado, que seja capaz de enviar uma mensagem de texto e que esteja localizado no mesmo ambiente onde se encontra o paciente (informação obtida na interação anterior). Esta consulta é similar à apresentada no Código 38 da Seção 5.1.3.6.

O Código 51 apresenta uma possível resposta, que retorna informações sobre o recurso registrado através da especificação contida no Código 39 da Seção 5.1.4.2.

```
1 <Response>
2   <ResourceInfo id="TV_LivingRoom" componentType="Sensor" capacityName="ActiveDevice">
3     <Attribute name="powerOn" value="true" unit="Boolean" type="Dynamic"/>
4   </ResourceInfo>

5   <ResourceInfo id="TV_LivingRoom" componentType="Actuator" capacityName="TextOutput">
6     <InvokeMethod invokeString="http://localhost:1988/TVAgent_LivingRoom?wsdl"
       operationName="writeMessage" returnType="string">
7   </ResourceInfo>
8 </Response>
```

Código 51 - Exemplo de resposta com um equipamento de TV

As Linhas 2-4 do Código 51 apresentam as características retornadas sobre o sensor que compõe o equipamento de TV e as Linhas 5-7 apresentam as características retornadas sobre seu atuador. Pode-se observar que para o sensor, foi retornado também o atributo *powerOn*, assumindo que este foi o termo utilizado para a consulta como no Código 38 da Seção 5.1.3.6. Já para o atuador, foram retornadas as informações contidas na Ontologia de Recursos sobre a operação que permite o acesso à funcionalidade buscada.

A partir dos dados do atuador do equipamento localizado, a aplicação pode então enviar o aviso necessário. Para isto ela executa a operação provida pelo Agente de Recurso que implementa este equipamento (Interação 5 da Figura 22).

Para finalizar, a aplicação inicia o monitoramento sobre a medição da pressão sanguínea, através do SC (Interação 6.1 da Figura 22), utilizando as referências obtidas no primeiro passo, através do SD. Quando os valores esperados são obtidos, a aplicação dá finalmente como concluído este passo no Plano de Cuidados.

5.1.6 Detalhes de Implementação

Para a implementação do protótipo foi utilizada a linguagem de programação Java SE versão 6 e o ambiente de desenvolvimento integrado (IDE na sigla em inglês) Netbeans (ORACLE3, 2011) versão 6.9. A criação do protótipo envolveu a criação de uma aplicação cliente, dos agentes de recursos e a definição de suas ontologias além dos simuladores para os dispositivos monitorados e utilizados na aplicação de telemedicina.

A IDE Netbeans foi escolhida por dar suporte nativo ao gerenciador de *layouts Matisse* que facilita o desenvolvimento de aplicações gráficas *desktop* com *Java Swing* (ORACLE4, 2011). O *Swing* é parte do Java SE e provê um conjunto de componentes para interface gráfica. O *Matisse* permite a criação de interfaces arrastando e posicionando componentes na tela. Enquanto o programador trabalha com os componentes disponíveis em um formulário, a IDE infere as definições de espaçamento, alinhamento e tamanho, entre outras.

Para a criação da imagem da residência do paciente monitorado foi utilizada a ferramenta *Google SketchUp* (GOOGLE1, 2011), que permite a criação de imagens e modelos 2D e 3D. Esta ferramenta foi utilizada por permitir a criação de um modelo realístico do ambiente envolvido na aplicação de telemedicina.

A imagem utilizada no protótipo foi obtida na biblioteca de exemplos do SketchUp e posteriormente alterada utilizando esta mesma ferramenta, disponibilizada de forma gratuita para uso privado e não comercial.

5.2 **Outras Aplicações**

Nesta seção serão apresentadas idéias para outras duas aplicações com base na infraestrutura desenvolvida neste projeto. Estas aplicações não serão detalhadas como feito para a aplicação de telemedicina.

5.2.1 Aplicação para Comunicação de Dados Através de Dispositivos Móveis

Grande parte da população brasileira economicamente ativa possui mais de um dispositivo móvel com acesso à Internet, comumente através de redes de operadoras de

telefonia celular diferentes, o que dá a este usuário mais de uma opção para a escolha do serviço de conexão de dados. A Agência Nacional de Telecomunicações (Anatel) divulgou no final de 2010 que o Brasil já ultrapassou a marca de mais de um celular por habitante (ANATEL, 2011). Também foi constatado que o número de acessos de banda larga móvel no Brasil já ultrapassou o de banda larga fixa.

Um usuário de telefonia celular pode possuir à sua disposição então diversos serviços para a comunicação de dados providos por uma ou mais operadoras. Estas operadoras podem aplicar tarifas diferentes para este serviço e provê-lo com qualidade diferenciada de acordo com condições pré-estabelecidas com seus clientes. Muitos aparelhos celulares também permitem a comunicação à Internet utilizando *wifi*, além da rede da operadora.

Além disso existem aparelhos no mercado com suporte a mais de um *SIM Card*, entre eles aparelhos de marcas menos tradicionais como o MP20 BOSS e de grandes empresas como a Nokia, com o seu modelo C2-00 e a Samsung que já disponibiliza vários modelos, como o B5702, D880 e o D980. Estes aparelhos permitem a utilização de serviços de mais de uma operadora, sem que seja necessário qualquer tipo de reinicialização do aparelho para o intercâmbio entre estes serviços. A utilização deste tipo de aparelho abre ainda mais as possibilidades para a escolha sobre os serviços de conexão de dados.

A decisão de um usuário sobre qual operadora utilizar a cada momento pode depender de diversos fatores, entre eles: preços cobrados pelo serviço, qualidade, disponibilidade e preferências pessoais.

Alguns fatores podem ser utilizados para a tarifação deste tipo de serviço, como: o plano de preços do assinante e se este possui ou não um pacote contratado, além da data e horário de utilização. Além disso, os pacotes de conexão de dados podem possuir limitações em termos de garantia de banda ou preço por MB, que podem ser aplicadas após certa quantidade de uso (uma certa cota de tráfego). Por exemplo, uma operadora pode reduzir a velocidade de conexão garantida para um assinante de 1Mbps para 256Kbps após o mesmo ter ultrapassado o limite de franquia de seu pacote, 8GB por exemplo, ao longo de um mês. Ou ainda, a operadora pode passar a cobrar uma tarifa superior por MB utilizado a partir do estouro desta cota de tráfego.

Muitos aparelhos celulares são capazes de selecionar entre o uso de *wifi* e a rede da operadora celular de forma automática. Porém esta escolha é simplificada. Não é disponibilizado um serviço que considere todas as características e possibilidades que podem

afetar a seleção de tais serviços e também não é possível a escolha entre mais de uma operadora, considerando de forma automática os preços praticados por cada uma delas, por exemplo.

A partir da existência de tantas variáveis para a escolha do melhor serviço para conexão de dados, dentre os disponíveis para um usuário, uma aplicação para auxiliar em tal decisão se torna útil para o mesmo.

Tal aplicação precisa ter acesso a diferentes tipos de informação. Algumas precisam ser fornecidas pelo usuário, como informações sobre as operadoras, planos e pacotes associados a cada *SIM Card*. Outras informações precisam ser obtidas a partir do contexto do usuário, como dia da semana, horário, disponibilidade de cada rede e a quantidade já utilizada dentro de cada pacote dentro do mês corrente, e ainda outras informações precisam ser obtidas sobre os serviços prestados pelas operadoras de telefonia.

A utilização do OCDRF em tal aplicação provê suporte à aquisição das informações de contexto necessárias e o suporte semântico necessário para a análise e decisão sobre as opções, com base nas informações obtidas. É proposta então uma aplicação para decisão sobre a melhor forma para conexão de dados para um usuário de telefonia celular que utiliza o OCDRF como infraestrutura.

A aplicação proposta utiliza critérios que envolvem as preferências definidas pelo usuário e critérios de desempate envolvendo qualidade de serviço e valor a ser pago. Para isto é necessária uma análise sobre as características das interfaces disponíveis.

As informações sobre as operadoras disponibilizadas para utilização da aplicação devem ser organizadas em forma de ontologias, de forma que possam ser utilizadas nas consultas através do OCDRF. São necessárias ontologias contendo informações sobre os preços de cada plano de serviços, por horário e dia da semana, além da identificação de feriados, visto que os serviços podem variar de preço nestas datas.

Cada forma disponível de conexão de dados deve se registrar junto à infraestrutura como um Agente de Recurso e consequentemente deve possuir uma descrição de acordo com a Ontologia de Recursos descrita neste trabalho. Cada forma disponível para a conexão de dados deve ser representada como um *Resource* onde devem ser descritos atributos como preço por MB, qualidade de serviço (dentro e fora da cota de seus planos de dados), etc.

As consultas ao Serviço de Descoberta do OCDRF passam então a ter acesso às informações registradas para representar cada forma de conexão de dados.

A aplicação proposta pode ser executada, por exemplo, sobre o sistema operacional da Google (GOOGLE2, 2011) para dispositivos móveis, Android (ANDROID, 2011). Este sistema permite que uma aplicação se encaixe ou sobrescreva funcionalidades do telefone como: interface de *email*, envio de SMS, teclado customizado, etc.

Fabricantes como a Motorola, LG e Samsung (WIKIPEDIA, 2011) já adotam este sistema operacional, que possui como característica interessante sua versatilidade. Ele pode ser utilizado em muitos tipos de dispositivos, entre eles: *Smart phones*, *Tablet computers*, *Netbooks*, *E-Readers*, TVs e em carros.

5.2.2 Aplicação Distribuída para Controle de Aquisição de Suprimentos

Uma necessidade muito comum em empresas e indústrias de médio e grande porte é o suporte à atividade de compras de insumos. Redes de varejo ou de restaurantes, por exemplo, podem se beneficiar de uma aplicação para este fim, agregando agilidade em seus negócios.

Esta questão é tão importante e de tamanho vulto para certas empresas, que negócios se formam apenas para gerir as compras de insumos para o seu funcionamento, realizando a ponte entre compradores e vendedores.

Neste ambiente, a utilização do OCDRF pode ser de grande valia para auxiliar nos processos de descoberta e compra de mercadorias. É então proposta uma aplicação distribuída baseada no OCDRF para controle de aquisição de suprimentos.

Para que a aplicação de controle de aquisição de suprimentos proposta seja capaz de localizar os insumos desejados, as informações sobre os produtos disponíveis para cada um dos fornecedores destes produtos devem estar disponíveis através do Serviço de Registro e Diretório do OCDRF.

Cada fornecedor deve ser representado junto à infraestrutura através de um Agente de Recurso e consequentemente cada fornecedor deve possuir uma descrição associada que deve estar de acordo com a Ontologia de Recursos descrita neste trabalho.

A empresa compradora, que utiliza a aplicação de controle de aquisição de suprimentos é responsável por disponibilizar os serviços da infraestrutura para seus fornecedores. Para que seja efetuado o registro de um fornecedor, este precisa conhecer o caminho para o Serviço de Registro e Diretório provido pela infraestrutura e diponibilizado

pela empresa compradora. Esta pode por exemplo, disponibilizar em seu site as informações necessárias para o acesso ao SRD para que os distribuidores se cadastrem.

Um fornecedor pode se registrar junto a mais de um cliente, provendo sua listagem de produtos disponíveis. Cada produto pode ser representado como uma *Capacity* dentro da Ontologia de Recursos. Esta irá possuir *Attributes* que irão representar informações como prazo para a entrega de tal produto, localização de seus distribuidores, quantidade em estoque, preço, desconto associado, garantia, possibilidade de troca e quantos mais forem necessários para a especificação dos produtos.

A Figura 23 apresenta como cada fornecedor irá se registrar junto ao *framework*.

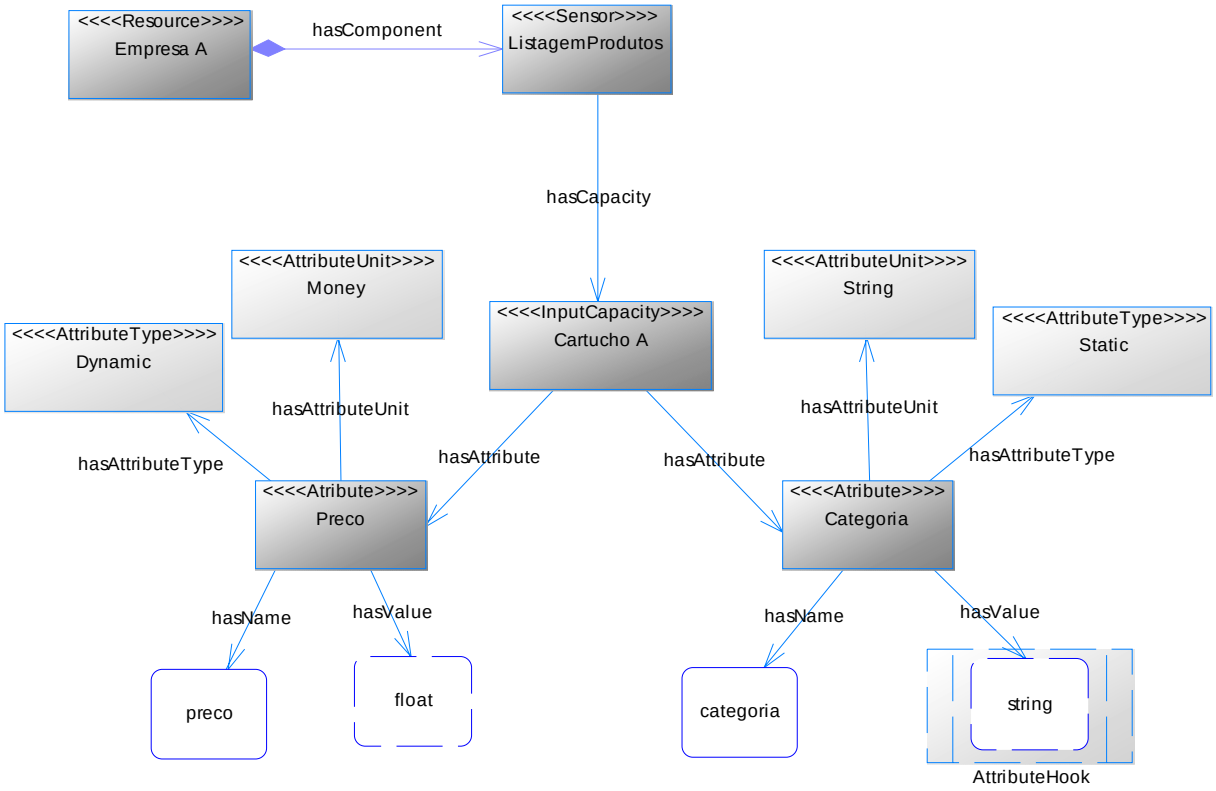

Figura 23 - Representação de empresas fornecedoras de mercadorias

A Figura 23 apresenta parte das instâncias da Ontologia de Recursos necessárias para descrever um fornecedor e um produto.

Muitas das informações detalhadas através dos atributos dos produtos fornecidos por cada vendedor, como preço (apresentado na Figura 23) e quantidade em estoque, são de natureza dinâmica e devem então ser providos pelos ARs de cada fornecedor como atributos dinâmicos a serem capturados através do Serviço de Contexto e Serviço de Descoberta.

Os ARs que representam os fornecedores de produtos podem também prover atuadores que suportem operações de compras, reservas e devoluções de mercadorias para que a aplicação de controle de aquisição possa efetuar as transações necessárias.

A partir do registro de fornecedores o sistema de controle de aquisição de suprimentos poderá então realizar suas escolhas com base nos produtos e quantidade que necessita, prazo de entrega de cada fornecedor, preços, grau de confiabilidade de cada fornecedor e descontos sobre quantidade solicitada, entre outros fatores, utilizando para isto os serviços de Contexto e Descoberta disponibilizados pela infraestrutura.

A aplicação proposta pode também se utilizar de outras ontologias (Ontologias Cliente) às quais os vendedores podem utilizar para a descrição de seus produtos.

Pode ser desenvolvida por exemplo uma ontologia para classificação de produtos. Esta ontologia pode conter elementos que indiquem a qualidade do produto, sua aceitação no mercado, sua categoria dentro de uma linha de produtos, detalhes sobre sua composição, informações sobre assistência técnica, etc.

Estas ontologias podem então ser acopladas ao OCDRF através de seus pontos de extensão e os termos definidos nestas ontologias podem então ser usados pela aplicação em suas consultas ao Serviço de Descoberta da infraestrutura.

6 TRABALHOS RELACIONADOS

Neste capítulo são apresentados trabalhos e propostas relacionadas. Os trabalhos selecionados apresentam infraestruturas para desenvolvimento de sistemas cientes de contexto que utilizam, assim como o OCDRF, ontologias na modelagem de contexto da aplicação para prover capacidade semântica ao processo de descoberta de recursos. Estas infraestruturas visam prover serviços para ambientes de estrutura dinâmica, como o de aplicações cientes de contexto, onde recursos podem se associar e se desligar da estrutura a qualquer momento. Tais soluções utilizam ontologias em seus serviços de acesso aos dados de contexto e em seus serviços de descoberta de recursos, durante o processo de casamento entre as informações de registro de tais recursos e as informações solicitadas pelas aplicações.

Serão discutidos aspectos relevantes da utilização de ontologias nas diferentes abordagens e como as ontologias são utilizadas e mantidas. Serão apresentados os pontos positivos e negativos de cada solução, e como a proposta da presente dissertação se posiciona em relação às mesmas.

6.1 Utilização de ontologias em serviços de contexto e descoberta

6.1.1 CxFramework

O *CxFramework* (TOSIN, 2009) é uma infraestrutura para ser utilizada por aplicações dependentes de contexto que integra suporte a contextos à plataforma CORBA (OMG, 2011). A modelagem das informações de contexto e a inferência de contextos de alto nível nesta infraestrutura é feita através do uso de ontologias.

Os quatro serviços disponibilizados pelo *CxFramework* foram criados como objetos CORBA registrados no CORBA *Naming Service*. São eles:

- *Sensor Management Service* → Serviço responsável pela comunicação entre a infraestrutura e os sensores externos. Ele é composto por *sensor adapters* que fazem o papel dos ARs em nossa infraestrutura, encapsulando a comunicação com os sensores.

- *Registry Service* → Serviço responsável por armazenar informações sobre os *sensor adapters* e sobre os eventos de mudança de contexto que a infraestrutura é capaz de gerar.
- *Context Reasoning Service* → Serviço que recebe e processa os eventos vindos dos *sensor adapters*, transformando estes eventos em informações de contexto de mais alto nível através do uso de ontologias. Quando um sensor detecta alguma alteração no contexto, esta informação é enviada para *o Context Reasoning Service* que atualiza a ontologia.
- *Context Notification Service* → Serviço responsável pela notificação de eventos de contexto para as aplicações que registram o interesse nestas informações. Este serviço pode funcionar nos modos *push* e *pull*, assim como o Serviço de Contexto em nossa infraestrutura.

O serviço fornecido pelo *CxFramework* funciona da seguinte forma: Os *sensor adapters* capturam novas informações dos seus sensores e mandam tais informações para o *Context Reasoning Service*. Este infere novas informações que são enviadas para o *Context Notification Service,* que se responsabiliza por notificar as aplicações clientes interessadas.

Ao contrário de como ocorre no OCDRF, o registro dos *sensor adapters* na infraestrutura não é automático, ele fica a cargo do administrador do *CxFramework*. O registro é feito através da configuração manual de um arquivo XML que é lido na inicialização do *Sensor Management Service*. Os eventos, aos quais as aplicações podem se registrar para obter informações de contexto atualizadas, precisam também ser cadastrados manualmente pelo administrador do sistema.

A ontologia e as regras de inferência precisam ser configuradas estaticamente para serem usadas no *Context Reasoning Service*. Cada *sensor adapter* precisa ser associado, também de forma manual, ao indivíduo e propriedade correspondentes na ontologia para que possa ser acessado. Isto é efetuado através da configuração da *event table*. É por meio da consulta nessa tabela que o serviço consegue definir qual parte da ontologia será alterada após uma notificação recebida de um *sensor adapter*.

A ontologia usada pelo *CxFramework* é usada apenas para interpretar as informações coletadas dos sensores e transformá-las em informações de mais alto nível, para que sejam passadas para as aplicações clientes. Ela não é responsável por armazenar a estrutura e características dos próprios sensores. Ao contrário do OCDRF, não há funcionalidade de

registro nem de buscas por recursos do contexto de execução das aplicações usando informações de contexto, com o suporte de ontologias.

O *CxFramework* possui ainda um *proxy* que precisa ser importado e utilizado nas aplicações clientes. Este *proxy* tem a finalidade de esconder detalhes relativos ao serviço de notificação do CORBA. Ao contrário, o OCDRF apenas exige que as consultas das aplicações clientes sejam realizadas utilizando o formato de mensagens definido, sendo menos intrusivo nas aplicações clientes.

6.1.2 CASD

O artigo (PAWAR, 2006) apresenta o serviço CASD (*Context-Aware Service Discovery*), capaz de determinar o serviço mais adequado para ser utilizado por uma aplicação em um ambiente sensível a contexto, levando em conta as informações de contexto do serviço e da aplicação cliente. Quando ocorre uma mudança de contexto ou o aparecimento de novos serviços, o CASD age de forma pró-ativa. Se um serviço mais adequado é encontrado, o cliente é notificado por meio de um serviço de descoberta persistente.

O OCDRF não fornece tais características. Ele não analisa os recursos retornados de forma a responder para a aplicação qual recurso melhor se aplicaria a ela, de acordo com a pesquisa recebida. Uma listagem com todos os recursos que se enquadram com os atributos da pesquisa é retornada, juntamente com os valores correntes de seus atributos, de forma que a aplicação cliente é responsável pela decisão de qual destes recursos irá utilizar. O OCDRF também não suporta o registro de *callback interfaces* no Serviço de Descoberta, como é provido pelo CASD.

No CASD as fontes de contexto foram modeladas como serviços que possuem funções padronizadas para a atualização e recuperação de informações de contexto, assim como os Agentes de Recurso de nossa proposta. Uma fonte de contexto é um serviço que fornece informações de contexto do serviço ou cliente associado.

No modelo conceitual para o CASD cada serviço e cliente pode ter uma ou mais fontes de contexto. Os serviços e as fontes de contexto precisam se registrar no serviço de diretório para que possam ser descobertos.

A utilização do CASD é efetuada como a seguir: um cliente solicita um serviço para o CASD. Este recupera os serviços que correspondem ao tipo de serviço especificado pelo

cliente, após consultar o serviço de diretório. Tais serviços são referidos pelo CASD como serviços de correspondência básica, que são serviços que poderiam ser retornados por um serviço de descoberta comum, ou seja, por um serviço de descoberta que não considere informações de contexto.

Após a descoberta dos serviços de correspondência básica, o serviço CASD coleta as informações de contexto destes serviços e do cliente. Com base nestas informações, o CASD é capaz então de filtrar os serviços para retornar o serviço mais adequado para o cliente.

Assim como no OCDRF, no CASD as ontologias são utilizadas para a representação de informações de contexto e para seu processamento durante o casamento de serviços. As ontologias fornecem um vocabulário comum para a especificação de informações de contexto do cliente e do serviço e as consultas ao conjunto de serviços estão intimamente ligadas a elas.

6.1.3 CHIL

No artigo (PANDIS, 2005) é apresentado um *framework* para serviços de computação sensível ao contexto no escopo de ambientes *in-door*, criado para suportar o projeto CHIL (*Computers in the Human Interaction Loop*). A ênfase deste *framework* está nos mecanismos de descoberta, invocação e gestão dinâmica dos recursos (serviços, equipamentos, redes, sensores e atuadores). Ele utiliza uma base de conhecimentos para descobrir e executar componentes dinamicamente e utiliza ontologias para tratar os desafios relacionados à integração dos componentes.

Os diversos agentes do *framework* podem ser classificados em agentes do núcleo básico, agentes de serviços básicos e agentes de serviços cientes de contexto. Os agentes do núcleo básico fornecem um mecanismo de comunicação para as entidades distribuídas do sistema, um conjunto de serviços básicos que fazem o controle da infraestrutura instalada e permitem que os prestadores de serviços conectem lógica de serviço ao *framework*.

Os agentes do núcleo básico incluem o *Situation Watching Agent* que provê uma funcionalidade similar à funcionalidade de registro de solicitações para notificações assíncronas, disponibilizada pelo Serviço de Contexto do CDRF (RODRIGUES, 2009a) e o *Agent Manager* que faz o papel equivalente ao do Serviço de Descoberta do CDRF.

O *Agent Manager* age como um “casamenteiro”. Quando recebe um pedido de serviço a partir de um outro agente, ele descobre através da base de conhecimentos qual agente de serviço pode resolver o problema e quais os recursos necessários para isso.

Os agentes de serviços básicos estão ligados aos agentes do núcleo básico e expõem suas capacidades através do mecanismo de gestão das ontologias. Este mecanismo está no coração do *framework* e é responsável por fornecer o mecanismo de registro; fornecer a solução de armazenamento permanente de dados das aplicações; permitir a intercambialidade entre os diversos componentes desenvolvidos por empresas diferentes e facilitar a comunicação entre os diversos agentes do sistema. Para esta última função, parte da ontologia é usada para modelar o tipo e o valor do conteúdo das mensagens que são trocadas entre os diversos agentes do sistema. Em resumo, assim como no OCDRF, o registro, descoberta, manipulação e integração dos recursos do sistema são realizados utilizando ontologias.

A ontologia criada visa estabelecer um vocabulário para os diversos conceitos dentro de um espaço multissensor inteligente, onde as aplicações sensíveis ao contexto são executadas. A principal finalidade desta ontologia é facilitar a pesquisa de componentes, descrevendo o tipo de informação que fornecem (por exemplo, as coordenadas de uma pessoa).

Com o uso da ontologia desenvolvida, componentes específicos da instalação do sistema podem ser registrados. Assim como a Ontologia de Recursos do OCDRF, a ontologia CHIL possui informações sobre os sensores e atuadores do contexto de execução das aplicações, entre elas, seu fornecedor, modelo, interfaces, capacidades e os meios de acesso aos seus recursos (isto é, sua API).

A ontologia CHIL possui ainda a informação sobre o estado atual dos dispositivos. O monitoramento e controle dos sensores e atuadores é feito através de agentes *proxy* que os representam. Um agente *proxy* registra o dispositivo na base de conhecimento após a inicialização do mesmo e atualiza este registro sempre que o estado do dispositivo é modificado.

No OCDRF optou-se por não atualizar a Ontologia de Recursos a cada atualização de estado dos sensores do contexto de execução das aplicações. Sempre que uma informação dinâmica precisa ser acessada, o Serviço de Contexto adquire junto ao Agente de Recurso adequado esta informação atualizada. Isto evita atualizações constantes sobre a Ontologia de Recursos com informações que podem ou não ser necessárias para as aplicações clientes.

6.1.4 COSS

O artigo (BROENS, 2004) apresenta uma abordagem para a descoberta de serviços que utiliza ontologias para capturar a semântica da consulta do usuário, dos serviços e das informações de contexto que são consideradas relevantes no processo de descoberta.

As entradas do componente de descoberta são: a consulta do usuário (ou seja, a solicitação de serviços), o conjunto de serviços registrados (isto é, as descrições de serviços), o conjunto de provedores de contexto e as ontologias utilizadas pelos usuários, pelos serviços e pelos provedores de contexto. O resultado do algoritmo de descoberta é um conjunto de descrições de serviços que correspondem semanticamente à consulta do usuário.

Os usuários, os provedores de serviços e os provedores de contexto alcançam um entendimento comum através de ontologias que todos compartilham. Os usuários e provedores de serviços possuem provedores de contexto associados, que podem oferecer diferentes tipos de informação de contexto.

Quatro propriedades dos serviços são tratadas pelo algoritmo de descoberta: o tipo do serviço, suas entradas, suas saídas e o atributo contextual. Os três primeiros representam conceitos que devem possuir correlações com ontologias e o último representa a informação de contexto derivada sobre o usuário, por exemplo sua localização, e sobre os provedores de serviços. Os atributos contextuais são associados a preferências que os usuários podem definir sobre certas propriedades do serviço que eles querem descobrir. Por exemplo, o usuário pode querer um serviço que esteja próximo a ele.

A abordagem definida pelo COSS foi incorporada ao componente *matchmaker* da plataforma WASP (FREEBAND, 2004), que fornece um ambiente de suporte para aplicações móveis cientes de contexto.

Ao contrário do COSS, no OCDRF as entradas e saídas das operações providas pelos componentes dos recursos são representadas por tipos simples, que não possuem correlação com outras ontologias e o tipo de um recurso é definido pela classe a qual tal recurso pertence e pelas correlações desta classe dentro da Ontologia de Recursos.

6.1.5 Gaia

No artigo (RANGANATHAN, 2003) é proposto um *middleware* para que agentes em um ambiente ciente de contexto sejam capazes de captar e raciocinar sobre o contexto atual do ambiente e interagir com outros agentes. Este *middleware* utiliza CORBA para a comunicação entre os agentes, e ontologias para descrever diferentes tipos de contextos. Ele também fornece vários serviços e bibliotecas para permitir a aquisição de agentes e o raciocínio sobre as informações de contexto.

Foi desenvolvido um modelo de contexto baseado em predicados. As ontologias são utilizadas para descrever as propriedades e estrutura dos diferentes predicados. Exemplo de predicado de contexto: “Location(chris, entering, room3231)”, que segue a estrutura “Tipo de Contexto”(<Sujeito>,<verbo>,<objeto>). Cada tipo de contexto corresponde a uma classe na ontologia.

A ontologia define os vários tipos de contexto assim como os argumentos que os predicados devem possuir e é usada para checar a validade dos predicados de contexto. Ela foi escrita em DAML + OIL (HARMELEN, 2001) e o FaCT *reasoning engine* (HORROCKS,1998) é usado para verificar a validade das expressões de contexto.

Este *middleware* foi integrado ao Gaia, que é uma infraestrutura para espaços inteligentes. O Gaia oferece maneiras diferentes para que os agentes possam adquirir diversos tipos de informações de contexto e raciocinar sobre elas.

Existem diferentes tipos de agentes que estão envolvidos na estrutura do Gaia, entre eles: *Context Providers*, *Context Synthesizers*, *Context Consumers*, *Context Provider Lookup Service* e *Ontology Server*.

Os *Context Providers* são equivalentes aos ARs da nossa infraestrutura. Os *Context Synthesizers* são responsáveis por deduzir informações de mais alto nível a partir de informações adquiridas dos *Context Providers*. Os *Context Synthesizers* suportam a especificação de novas regras de inferência para a dedução de novas informações de contexto.

Context Consumers são as aplicações sensíveis a contexto. Elas podem adquirir diferentes tipos de contexto a partir dos *Context Providers* ou dos *Context Synthesizers*.

O *Context Provider Lookup Service* permite que agentes encontrem *Context Providers*. Os *Context Providers* anunciam o conjunto de contextos que fornecem para o

Context Provider Lookup Service e o *Ontology Server* mantém as ontologias que descrevem os diferentes tipos de informações de contexto.

Um aplicativo que deseje saber informações de contexto envia uma consulta ao *Context Provider Lookup Service*. Este serviço busca por um provedor de contexto que forneça o contexto que a aplicação está interessada e retorna uma referência deste *Context Provider* para a aplicação.

Ao contrário do OCDRF, o Gaia não utiliza ontologias para auxiliar no processo de descoberta. A descoberta é feita através do *Context Provider Lookup Service* e depois os agentes podem buscar as ontologias que lhe interessam no *Ontology Server*.

Foram também desenvolvidas ontologias que descrevem diferentes tipos de agentes e suas propriedades, além de ontologias que descrevem a estrutura para as informações de contexto. As ontologias são mantidas pelo servidor de ontologias e os agentes entram em contato com este para obter as descrições dos agentes, metainformações sobre o contexto e definições de diversos termos utilizados pelo Gaia.

O servidor de ontologias fornece uma interface para adicionar novos conceitos nas ontologias existentes a qualquer momento durante o funcionamento do sistema. Ele garante que quaisquer novas definições sejam logicamente consistentes com as definições existentes.

Aplicações clientes podem obter a estrutura dos contextos em que estão interessadas a partir do servidor de ontologias. Elas podem então preparar consultas adequadas para os *Context Providers* para obter os contextos que necessitam.

O Gaia também auxilia no tratamento de heterogeneidade de definições dos termos entre os agentes. O uso de ontologias permite que agentes em diferentes ambientes se comuniquem. Para isto, precisa ser feito o mapeamento dos conceitos dos dois ambientes através de ontologias.

Cada agente no ambiente utiliza o vocabulário e os conceitos definidos em uma ou mais ontologias. Quando dois agentes diferentes “falam” um com o outro, eles sabem qual ontologia o outro agente usa e conseguem assim compreender a semântica do que o outro agente está “dizendo”.

6.1.6 CSN

A arquitetura apresentada em (POWER, 2004) utiliza uma abordagem baseada em ontologia para lidar com a heterogeneidade das fontes de informação de contexto, e fazer a tradução e roteamento de consultas utilizando as informações das ontologias disponíveis.

Esta é a proposta que, em termos de funcionalidade e extensibilidade, mais se assemelha do trabalho proposto nesta dissertação, apesar de sua arquitetura ser bem diferente da definida para o OCDRF.

Neste artigo é descrito um serviço de contexto que responde a consultas com base em ontologias. Ele age como um mediador entre o cliente e outras fontes de informação.

O serviço de contexto é implementado por um conjunto de nós (*Context Service Nodes* - CSN). Este serviço recebe junto com uma consulta, uma referência à ontologia a qual a consulta se refere. Esta correlação não precisa ser feita no OCDRF no momento da consulta, dado que esta ligação com as ontologias é feita na inicialização dos serviços.

Um aplicação pode agir como cliente e fonte de informação de contexto. Para isto, ela deve definir uma ontologia que descreva o domínio das informações de contexto que ela está interessada em consultar e também sobre o que quer disponibilizar para outras aplicações e fornecer esta ontologia para o serviço de contexto. Fica garantido assim que o CSN onde tal ontologia foi registrada será capaz de compreender consultas vindas de suas aplicações. A ontologia é registrada no serviço de contexto como pertencendo à aplicação e armazenada em um repositório de ontologias, acessível ao serviço de contexto.

Combinando os nós (CSNs) com os mapeamentos do repositório de ontologias, o CSN (nó) que receber uma consulta pode então compor uma nova consulta que, por sua vez, pode ser encaminhada para outros CSNs, que tentarão retornar o resultado correspondente. Os resultados são então traduzidos da ontologia do CSN remoto para a ontologia do aplicativo cliente antes de serem devolvidos como resposta da consulta. Os termos envolvidos na consulta devem ser um subconjunto daqueles descritos nas ontologias entendidas pelo serviço de contexto, para que este seja capaz de entender a consulta.

Os serviços de Contexto e Descoberta do OCDRF realizam um tratamento similar sobre as respostas entregues à aplicação cliente. As mesmas são retornadas utilizando os termos utilizados originalmente pela aplicação cliente, poupando-a da necessidade de qualquer tipo de conversão de termos.

Associado a cada CSN existe um repositório de ontologias que descreve os domínios de conhecimento que o nó armazena. O CSN deve pegar cada consulta recebida e se comunicar com outros CSNs para resolvê-la.

Como no CSN não há um repositório único para as informações dos nós, não há como se obter uma visão global do contexto. Com o OCDRF seria possível extrair uma fotografia do ambiente, para manter histórico, ou analisar algum comportamento momentâneo, através da leitura sobre apenas um serviço; ao contrário do CSN, que possui suas informações pulverizadas.

Quando uma consulta é recebida e analisada, o primeiro CSN deve decidir para onde esta consulta deve ser encaminhada. Para isto, o CSN pode utilizar um algoritmo de roteamento. No caso mais simples, a consulta será roteada a um segundo nó que irá compreendê-la (pois possui os termos equivalentes em suas ontologias locais) e será portanto, capaz de devolver uma resposta significativa. Esta consulta pode ser compreendida por qualquer CSN que possua termos equivalentes em suas próprias ontologias locais. Uma consulta também pode ser enviada via *broadcast*, mas isto pode ser ineficiente em redes grandes.

As abordagens para o reenvio da consulta para outros nós possuem benefícios diferentes. Consultas enviadas a um número pequeno de nós são eficientes em termos de custo de banda e processamento porém consultas enviadas a um número maior de nós podem ter mais chances de retornar satisfatoriamente. Também deve ser tratada a situação em que um grande número de nós puderem compreender uma consulta e várias respostas sejam retornadas. Na arquitetura adotada para o OCDRF não é necessário este tipo de tratamento.

6.1.7 AMIGO

O projeto AMIGO usa ontologias para diversas finalidades. (RAMPARANY, 2006) e (LEUTNANT, 2007) focam na modelagem/representação e compartilhamento de informações de contexto, além de raciocínio sobre elas utilizando ontologias. Já em (LACOSTE, 2007), ontologias são usadas para classificar a “reputação” e a confiabilidade das informações de contexto vindas de diferentes fontes.

Em (RAMPARANY, 2006) é apresentada uma infraestrutura para gestão de informações de contexto, detalhando o CMI (*Context Management Infrastructure*). A gestão

de contexto demanda adquirir dados de várias fontes, tais como sensores físicos, atividades do usuário ou aplicações em execução, e combinar estas peças de dados heterogêneos em informações consistentes que são fornecidas para as aplicações sensíveis ao contexto. Uma ontologia é utilizada para identificar os tipos de informações de contexto relevantes para as aplicações sensíveis ao contexto e as relações entre estes tipos.

Assim como utilizado no registro de recursos no OCDRF, um formato textual serializado baseado em XML é utilizado no AMIGO para transmitir informações entre as fontes de contexto e aplicações sensíveis ao contexto. (AMIGO, 2011) apresenta documentos OWL, transmitidos por compontes do AMIGO, que contêm a descrição semântica de itens de multimídia. Uma aplicação que deseje utilizar as informações contidas em um destes documentos OWL irá recuperar o documento, importar recursivamente as ontologias indicadas neste documento e tratar como desejar a descrição completa do item multimídia buscado.

Outro componente da arquitetura, o CMS (*Context Management Service*) *Context Broker* (LEUTNANT, 2007), fornece toda a funcionalidade necessária para se descobrir e usar fontes de contexto. O *Context Broker* realiza as funções de registro e descoberta, providas por serviços diferentes no OCDRF.

O AMIGO não estabelece um serviço que realize a comunicação entre um cliente e uma fonte de contexto, como o Serviço de Contexto do OCDRF. Um cliente precisa acessar diretamente as fontes de contexto que lhe interessam para obter informações atualizadas.

Para permitir que uma fonte de contexto seja descoberta pelos clientes, esta precisa estar registrada no *Context Broker*. O argumento para o registro é uma *string* que descreve as capacidades da fonte de contexto e é expressa em RDF. Porém estas informações não são utilizadas para casamentos semânticos, como os fornecidos pelo Serviço de Descoberta do OCDRF. Para que um cliente localize e possa acessar uma fonte de contexto, ele precisa utilizar uma *string* RDF idêntica à registrada pela fonte de contexto no *Context Broker*.

Na estrutura do AMIGO, a criação de um cliente de contexto está então diretamente conectada a criação de uma fonte de contexto. O cliente precisa estar ciente da fonte de contexto, do tipo de informação que ela provê, e principalmente da ontologia que ela está usando para representar as informações de contexto. Esta característica restringe totalmente os clientes de contexto, uma vez que a consulta do cliente de contexto tem de ser adaptada ao modelo criado pela fonte de contexto.

Finalmente, de posse da informação sobre a fonte de contexto a ser acessada, uma consulta pode ser enviada. Esta deve ter como argumento uma consulta expressa em SPARQL e terá como retorno outra *string* contendo a resposta da consulta. Uma fonte de contexto pode enviar informações para o cliente em modo "*query driven*" ou "*data driven*", isto é, *pull* ou *push*, sendo necessário para este último um registro por parte do cliente, assim como suportado pelos Agentes de Recurso disponibilizados pelo OCDRF.

6.2 **Resumo**

Todas as propostas apresentadas neste capítulo utilizam ontologias para a modelagem de informações de seu contexto.

Ao contrário de como ocorre no OCDRF, o registro das interfaces (equivalentes aos Agentes de Recursos) que encapsulam os sensores do ambiente no *CxFramework* é feito de forma manual. A ontologia usada no *CxFramework* não é responsável por armazenar a estrutura e características dos próprios recursos do ambiente e o *CxFramework* não fornece funcionalidade de registro nem de buscas de recursos do ambiente usando informações de contexto com o suporte de ontologias.

A ontologia usada no CASD também não é responsável por armazenar a estrutura e características dos próprios recursos do ambiente.

Dentre os trabalhos relacionados apresentados, apenas três suportam o registro de recursos através de sua descrição em OWL, sendo evitados nestas infraestruturas os passos extras para a extração e conversão de dados de registro.

Assim como no OCDRF, o registro, descoberta e manipulação dos recursos do sistema no CHIL e no COSS são realizadas utilizando ontologias.

Ao contrário do OCDRF, o Gaia e o AMIGO não utilizam ontologias para auxiliar no processo de descoberta, i.e., não fornecem funcionalidade de buscas de recursos do ambiente usando informações de contexto com o suporte de ontologias.

O *Context Broker* provido pelo AMIGO realiza as funções de registro e de descoberta, providas por serviços diferentes no OCDRF. Porém o processo de descoberta provido por ele não realiza casamentos semânticos. Para que um cliente localize e possa acessar uma fonte de contexto, ele precisa utilizar uma *string* RDF idêntica à registrada pela fonte de contexto no *Context Broker*.

O OCDRF exige apenas que as consultas recebidas das aplicações clientes sejam realizadas utilizando o formato de mensagens definido, sendo muito pouco intrusivo nas aplicações clientes. Por sua vez, o *CxFramework* fornece uma camada de *software* que precisa ser importada e utilizada nas aplicações clientes para que a complexidade de acesso a ele seja escondida.

Ao contrário do OCDRF, o CHIL mantém sua ontologia atualizada com o estado mais recentes dos recursos do contexto de execução das aplicações. No OCDRF optou-se por não atualizar a Ontologia de Recursos desta forma a cada atualização de estado dos sensores do ambiente. Sendo assim, sempre que uma informação dinâmica precisa ser extraída do contexto de execução das aplicações, o Serviço de Contexto precisa adquirir, junto ao Agente de Recurso adequado, esta informação atualizada. Isto evita atualizações desnecessárias na Ontologia de Recursos, visto que somente informações necessárias às aplicações clientes são extraídas, quando necessário.

Dentre as propostas analisadas, apenas o CSN disponibiliza pontos de extensão nativos para que as aplicações clientes possam utilizar ontologias que descrevam seu domínio de atividade, para que os termos definidos nestas ontologias possam ser utilizados nas buscas enviadas à infraestrutura e comparados semanticamente a informações de contexto dos recursos pesquisados.

O COSS permite que as entradas, as saídas e o tipo de um serviço sejam descritos através de ontologias, o que poderia ser considerado um ponto de extensão. Porém a utilização destas ontologias está restrita a estes pontos. Nenhuma outra característica destes serviços pode ser descrita através de ontologias cliente de forma a ser usada no processo de descoberta.

O AMIGO suporta a definição de novas ontologias por parte das fontes de contexto. Porém esta funcionalidade não é compreendida como um ponto de extensão comparável ao disponibilizado pelo OCDRF, uma vez que estas ontologias não são utilizadas no processo de descoberta de recursos do contexto de execução das aplicações.

Dentre os trabalhos pesquisados, o CSN foi a proposta que mais se assemelha ao trabalho proposto nesta dissertação, apesar de sua arquitetura ser bem diferente da definida para o OCDRF. Ao contrário do OCDRF, com o CSN não há como se obter uma visão global do contexto, uma vez que não há um repositório único para as informações. Além disso, o

CSN tem que lidar com questões de roteamento de consultas, que não são problemas para a arquitetura adotada para o OCDRF.

O serviço de contexto definido para o CSN precisa receber junto com uma consulta, uma referência à ontologia à qual a consulta se refere. Esta correlação não precisa ser feita no OCDRF no momento da consulta, uma vez que esta ligação com as ontologias é feita na inicialização dos serviços.

A Tabela 1 apresenta uma comparação das propostas analisadas e apresentadas neste capítulo para infraestruturas de suporte às aplicações cientes de contexto que utilizam ontologias na modelagem de contexto. Como último elemento nesta tabela é apresentado o CDRF, infraestrutura que serviu como base para a criação do OCDRF.

Tabela 1 - Comparação das propostas que utilizam ontologias na modelagem de contexto

	Registro automático das informações dos recursos	**Registro automático das informações dos recursos utilizando ontologias**	**Uso de ontologias como diretório para os recursos**	**Análises semânticas nos serviços de contexto**	**Pontos de extensão para suporte a ontologias externas**
CxFramework	Não	Não	Não	Não	Não
CASD	Sim	Não	Não	Sim	Não
CHIL	Sim	Não	Sim	Sim	Não
COSS	Sim	Sim	Sim	Sim	Não
Gaia	Sim	Não	Sim	Não	Não
CSN	Sim	Sim	Sim	Sim	Sim
AMIGO	Sim	Sim	Sim	Não	Não
CDRF	Sim	Não	Não	Não	Não

7 CONCLUSÕES

Este capítulo apresenta as conclusões sobre o trabalho realizado e propostas de extensões do mesmo.

7.1 Resultados Alcançados

Sistemas sensíveis ao contexto são, por natureza, interativos e envolvem múltiplos dispositivos e serviços. Eles provêem comportamentos apropriados, adaptando-se conforme seu ambiente de execução. Estes sistemas são por natureza abertos e extensíveis e precisam integrar informações de diversas fontes.

Um requisito importante para infraestruturas de suporte a sistemas sensíveis ao contexto é que estas permitam a interação entre agentes heterogêneos com o máximo de transparência possível. Apesar de um grande número de protocolos e *middleware* (como CORBA, Jini e SOAP) terem sido criados para habilitar a interação entre agentes distribuídos, estes mecanismos não resolvem questões de interoperabilidade sintática e semântica entre os agentes. Eles não provêem uma terminologia comum e um conjunto compartilhado de conceitos que os agentes possam usar quando estes interagem uns com os outros. Este problema é especialmente grave para o compartilhamento de informações de contexto, uma vez que diferentes agentes podem ter compreensões diferentes sobre o contexto atual. Eles podem usar termos diferentes para descrever o contexto, ou podem usar os mesmos termos porém com semântica diferente. Um *middleware* para suporte à sensibilidade a contexto deve resolver este problema, garantindo que não haja diferença semântica entre os termos usados pelos diferentes agentes, quando eles trocam informações de contexto. (RANGANATHAN, 2003)

As ontologias fornecem um mecanismo bem fundamentado para a representação e intercâmbio de informações estruturadas, provendo semântica às informações de contexto.

Através da utilização de ontologias é possível definir formalmente um conjunto comum de termos que são usados para descrever e representar um domínio do conhecimento. As ontologias capturam e especificam de forma geral, formal e explícita o conhecimento de um domínio com sua semântica intrínseca através de terminologias consensuais, axiomas formais e restrições. Ou seja, as ontologias envolvem um vocabulário bem formado com relações claramente definidas entre os diferentes termos do domínio.

A utilização de ontologias, como meio para o compartilhamento e reutilização de conhecimento, ganhou reconhecimento em vários campos da ciência da computação, incluindo comércio eletrônico e sistemas de apoio à decisão.

A utilização de ontologias para a descoberta de serviços em ambientes sensíveis a contexto apresenta diversas vantagens. As ontologias fornecem um vocabulário comum para a especificação de consultas do usuário, de descrições dos serviços e de informações de contexto. Este vocabulário fornece uma base para o casamento semântico entre as consultas dos usuários e as descrições dos serviços.

O presente trabalho apresentou o OCDRF, uma infraestrutura de suporte para aplicações cientes de contexto que utiliza ontologias para representar os recursos do ambiente de execução das aplicações, provendo consultas semânticas às informações sobre estes recursos.

Através da utilização do OCDRF, uma aplicação pode ter acesso a recursos de seu ambiente de execução sem a necessidade de utilizar o nome exato do recurso ou o nome exato do tipo deste recurso, como tenha sido registrado no sistema. Os serviços fornecidos pelo OCDRF permitem que consultas sobre o contexto da aplicação façam o casamento semântico entre os termos pesquisados e os termos registrados pelos serviços na infraestrutura.

Consultas que não realizam a análise semântica dos termos envolvidos podem ser frustrantes para o cliente, devido à limitação das técnicas de casamento entre palavras-chave. Os clientes frequentemente experimentam um dos dois problemas: consultas podem retornar sem resultados ou com muitos resultados irrelevantes.

O OCDRF provê um serviço de descoberta capaz de buscar por classes de recursos, providos por fontes diferentes, considerando suas capacidades e equivalências semânticas, não apenas seus nomes. Os atributos, que representam as restrições de tais consultas, também têm sua semântica analisada pelo OCDRF.

Para que o OCDRF proveja serviços com capacidade de análise semântica, o registro das informações dos recursos do contexto das aplicações é suportado por uma ontologia denominada Ontologia de Recursos.

As informações de registro dos recursos são definidas usando OWL e especificam instâncias de classes definidas na Ontologia de Recursos, e novas classes e propriedades, conforme a necessidade para a descrição detalhada dos recursos. Estas informações são

serializadas e transferidas para o Serviço de Registro e Diretório do OCDRF, que as acrescenta automaticamente na ontologia já registrada.

A Ontologia de Recursos possui a definição dos componentes, características e atributos dos recursos disponíveis no contexto de execução das aplicações clientes. Ela funciona no OCDRF como diretório para as informações dos recursos e como base para as extensões disponibilizadas para a utilização de outras ontologias. A Ontologia de Recursos permite a especificação dos comportamentos (capacidades) que podem ser esperados de cada recurso.

Os serviços de Contexto e Descoberta providos pelo OCDRF, além das buscas semânticas que executam, realizam também um tratamento sobre as respostas entregues à aplicação cliente. As mesmas são retornadas utilizando os termos utilizados originalmente pela aplicação cliente, poupando-a da necessidade de qualquer tipo de conversão de termos.

Este trabalho apresentou também os pontos de extensão disponibilizados pelo OCDRF com base na Ontologia de Recursos, que permite que conceitos das aplicações clientes sejam mapeados através de novas ontologias, que podem ser facilmente configuradas e utilizadas através da infraestrutura. Os pontos de extensão providos são um grande diferencial em relação aos trabalhos relacionados pesquisados.

Os termos envolvidos nas consultas devem ser um subconjunto daqueles descritos nas ontologias entendidas pelo Serviço de Descoberta, sejam eles da Ontologia de Recursos ou de outras ontologias, configuradas nos pontos de extensão da infraestrutura, para que este seja capaz de entender a consulta.

A linguagem utilizada neste trabalho para a criação de ontologias foi o OWL (W3C2, 2004) e o Jena (JENA, 2011) foi escolhido para manipular as ontologias e as regras de inferência definidas.

Foi realizada uma análise sobre o desempenho da infraestrutura proposta para o atendimento das consultas disponibilizadas. Foram feitas simulações utilizando os diversos tipos de consultas à infraestrutura, considerando uma proporção de utilização para cada consulta como espera-se que sejam executadas em uma aplicação real. Esta análise de desempenho mostrou que a infraestrutura desenvolvida é capaz de responder com qualidade, mesmo quando submetida a um alto nível de solicitações.

Três aplicações foram apresentadas neste trabalho para demonstrar como a infraestrutura proposta pode ser utilizada, provendo suporte a aplicações de diferentes

domínios. A proposta deste projeto foi então validada através do desenvolvimento de um protótipo para aplicações de telemedicina. Foram apresentadas as vantagens e flexibilidades acrescentadas a tais aplicações através da utilização do OCDRF.

7.2 Trabalhos Futuros

A partir da infraestrutura disponibilizada pelo OCDRF várias extensões podem ser desenvolvidas de forma a prover serviços melhores e de mais alto nível para as aplicações cientes de contexto. Serão descritas nesta seção algumas das idéias para trabalhos a serem desenvolvidos sobre o OCDRF.

O Serviço de Descoberta provido pelo OCDRF poderia realizar uma classificação e seleção dos recursos de forma a retornar para a aplicação uma lista ordenada priorizando os recursos que melhor se aplicariam, de acordo com a consulta recebida. Atualmente uma listagem com todos os recursos que se enquadram, de acordo com os critérios da pesquisa é retornada, juntamente com os valores correntes de seus atributos. De forma que a aplicação cliente é responsável pela decisão de qual destes recursos irá utilizar.

Outra extensão possível é adicionar a função de registro de interfaces para *callback* no Serviço de Descoberta, como é provido pelo CASD (PAWAR, 2006), de forma que ele possa responder a consultas no modo assimétrico. Isto traz novas possibilidades de utilização para as aplicações clientes, que poderiam ser informadas quando um recurso com um certo conjunto de características passasse a estar disponível.

Caso a funcionalidade de classificação entre os recursos do contexto de execução das aplicações fosse associada à funcionalidade de registro destas interfaces de *callback*, seria possível ainda ao Serviço de Descoberta informar à aplicação cliente quando um recurso melhor do que o provido anteriormente estivesse disponível. Desta forma, a aplicação cliente teria a seu dispor sempre o melhor dentre os recursos disponíveis em seu contexto de execução, segundo critérios da própria aplicação.

Além das traduções já efetuadas pelo Serviço de Contexto e pelo Serviço de Descoberta providos pelo OCDRF, para que a aplicação cliente receba em suas respostas os mesmos termos utilizados nas perguntas, ao invés dos termos registrados, pode também ser provida a conversão de formatos dos valores das respostas. Assim, as aplicações clientes também poderiam determinar o formato dos atributos que desejam receber, ao invés do

formato registrado pelos recursos. O formato das consultas aceitas pelo SC e pelo SD já está preparado para esta funcionalidade, bastando apenas que a mesma seja implementada.

O SC também poderia ser extendido para, assim como o SD, usar as ontologias definidas pelas aplicações clientes para o tratamento de suas consultas.

O OCDRF fornece a infraestrutura suficiente para o acesso às operações providas pelos sensores que compõem os recursos do contexto de execução das aplicações clientes, de forma a permitir que estas aplicações clientes obtenham os dados do contexto que sejam de seu interesse. Um trabalho futuro é complementar esta infraestrutura de forma a dar suporte para estas aplicações também para a execução das operações providas pelos atuadores que compõem estes recursos.

A descrição de um atuador e a configuração da operação que o implementa, já são providas pela Ontologia de Recursos. Basta para um trabalho futuro, a modelagem e o tratamento dos atributos de tais operações. Pode ser provido pela infraestrutura um novo serviço, para realizar a interface com os Agentes de Recurso com o objetivo de executar as operações solicitadas pela aplicação cliente. Ou ainda, esta nova tarefa pode ser executada diretamente pelas aplicações clientes, uma vez que elas já tenham acesso aos detalhes necessários para o acesso a tais operações.

Durante o desenvolvimento deste projeto, uma nova versão do OWL, denominado OWL 2 (W3C5, 2009), foi disponibilizada pelo W3C. Sendo assim, este trabalho pode ser adaptado para se tornar compatível com esta nova recomendação da W3C.

Em uma próxima etapa, um novo elemento pode também ser incorporado ao *framework*, que seja capaz de responder a consultas de agregação e a perguntas do tipo sim/não que facilitem ainda mais a criação de aplicações cientes de contexto. Este elemento agregador poderia, por exemplo, receber uma consulta para retornar se a temperatura média de um cômodo está acima de um determinado valor.

Para responder a esta questão, o elemento agregador se responsabilizaria por solicitar ao SC as informações atualizadas de todos os componentes disponíveis no cômodo especificado, que tenham a capacidade de medir a temperatura do ambiente. De posse dos dados monitorados, o elemento agregador seria então capaz de calcular a temperatura média e compará-la com o valor especificado na requisição.

Além das consultas de agregação, este elemento agregador poderia se beneficiar de ontologias descrevendo preferências, que poderiam ser enviadas dinamicamente pela

aplicação cliente. Por exemplo, uma aplicação poderia enviar para o elemento agregador uma ontologia com a definição do que é considerado uma temperatura agradável. Depois a aplicação poderia lhe fazer uma consulta como: “A temperatura do quarto principal está agradável?”. O elemento agregador usando suas habilidades de agrupamento conseguiria então calcular a temperatura média, e com base na ontologia recebida, responder a esta questão. Técnicas como redes neurais e lógica *fuzzy* também poderiam ser utilizadas neste elemento agregador para responder a questões de mais alto nível.

Como trabalho futuro, também podem ser desenvolvidas as aplicações apresentadas na Seção 5.2, além das propostas apresentadas nesta seção ligadas diretamente ao OCDRF.

REFERÊNCIAS

A&D. Disponível em: <http://www.andonline.com/medical/>. Acesso em: maio 2011.

AMIGO. Disponível em: <https://gforge.inria.fr/frs/download.php/2219/multimedia-ontologies-doc.zip>. Acesso em: jun. 2011.

ANATEL. Disponível em: <http://www.anatel.gov.br>. Acesso em: jun. 2011.

ANDROID. Disponível em: <http://www.android.com/>. Acesso em: jun. 2011.

APACHE SOFTWARE FOUNDATION. Apache Log4j 1.2.16 API. Disponível em: <http://jakarta.apache.org/log4j/docs/api/index.html>. Acesso em: jun. 2011.

APPLE. iPhone. Disponível em: <http://www.apple.com/iphone/>. Acesso em: jun. 2011.

BAUM, D. Semantic Breakthrough. Oracle Magazine - Maio/Junho de 2006, p. 42-46. Volume XX, Number 3. Disponível em: <http://www.oracle.com/technetwork/issue-archive/2006/06-may/o36semantic-083749.html>. Acesso em: maio 2011.

BONINO, D.; CORNO, F.; FARINETTI, L.; BOSCA, A. 2004 Ontology driven semantic search. WSEAS Transaction on Information Science and Application 1, 1597–1605.

BROENS, T. et al. Context-Aware, ontology based, service discovery. Proceedings of the European Symposium on Ambient Intelligence 2004 (EUSAI'04), LNSC 3295, Eindhoven, the Netherlands, 2004.

BROWNE, C. LGE to display Internet fridge at Gitex. Setembro, 2000. Disponível em: <http://www.arabianbusiness.com/lge-display-internet-fridge-at-gitex-143126.html>. Acesso em: maio 2011.

BULCÃO, R. Um processo de software e um modelo ontológico para apoio ao desenvolvimento de aplicações sensíveis a contexto. Doutorado/USP. 2006.

CARVALHO, S. T.; ERTHAL, M.; MARELI, D.; SZTAJNBERG, A.; COPETTI, A.; LOQUES, O. Monitoramento Remoto de Pacientes em Ambiente Domiciliar. In: XXVIII

Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - Salão de Ferramentas, SBRC 2010, Gramado, Rio Grande do Sul, Brasil, 2010.

COPETTI, A.; LOQUES, O.; LEITE, J.; BARBOSA, T.; DA NÓBREGA, A. (2009). Intelligent context-aware monitoring of hypertensive patients. In 1st Workshop for Situation Recognition and Medical Data Analysis. 3rd Int. Conf.on Pervasive Computing Technologies for Healthcare, London, UK.

DANG, S.; DIMMICK, S.; KELKAR, G. Evaluating the Evidence Base for the Use of Home Telehealth Remote Monitoring in Elderly with Heart Failure. Telemedicine and e-Health. Out. 2009, v.. 15, No. 8: 783-796.

DEY A. Providing architectural support for context-aware applications. [tese]. Georgia Institute of Tecnology; nov. 2000.

DISABLED LIVING FOUNDATION (DLF). What is telecare? Disponível em: <http://www.livingmadeeasy.org.uk/scenario.php?csid=54>. Acesso em: jun. 2011.

EKAHAU. Disponível em: <http://www.ekahau.com/>. Acesso em: jun. 2011.

FREEBAND. WASP project. 2004. Disponível em: <http://www.freeband.nl/kennisimpuls/projecten/wasp/ENindex.html>. Acesso em: maio 2011.

GENTIVA. Disponível em: <http://www.gentiva.com>. Acesso em: maio 2011.

GOOGLE1. Google SketchUp. Modelagem 3D para todos. Disponível em: <http://sketchup.google.com/>. Acesso em: jun. 2011.

GOOGLE2. Disponível em: <http://www.google.com/intl/en/corporate/>. Acesso em: jun. 2011.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies 43, 907–928, 1995.

GUARINO, N. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. SCIE, p. 139–170, 1997. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.53.939&rep=rep1&type=pdf>. Acesso em: jun. 2011.

HARMELEN, F. et al. Reference Description of the DAML+OIL ontology markup language. Março 2001. Disponível em: <http://www.daml.org/2001/03/reference.html>. Acesso em: maio 2011.

HEFLIN, J.; HENDLER, J. Semantic Interoperability on the Web. Proceedings of Extreme Markup Languages 2000, Ago. 2000, p. 1-2.

HORROCKS, I. The FaCT System. Automated Reasoning with Analytic Tableaux and Related Methods, 1998.

HOSPITAL-LAR. Disponível em: <http://www.hospital-lar.com.br>. Acesso em: maio 2011.

HP SEMANTIC WEB TEAM. Changes to Semantic Web research at HP. Out. 2009. Disponível em: <http://www.hpl.hp.com/semweb/>. Acesso em: maio 2011.

IDRISS, S. Z. Home telehealth: the future of home care. Disponível em: <http://www.mtbeurope.info/content/ft611003.htm>. Acesso em: maio 2011.

JAX-WS. GlassFish community. JAX-WS Reference Implementation. Disponível em: <http://jax-ws.java.net/>. Acesso em: jun. 2011.

JENA COMMUNITY. Jena – A Semantic Web Framework for Java. Disponível em: <http://www.openjena.org/>. Acesso em: jun. 2011.

KARPHOMECARE. Disponível em: <http://www.karphomecare.com>. Acesso em: maio 2011.

KORKEA-AHO, M. Context-Aware Applications Survey. Abril 2000. Disponível em: <http://www.cse.tkk.fi/fi/opinnot/T-110.5190/2000/applications/context-aware.html>. Acesso em: jun. 2011.

LACOSTE, M.; PRIVAT, G.; RAMPARANY, F. Evaluating confidence in context for context-aware security. Ambient Intelligence. LNCS; nov. 2007; v. 4794. p. 211-229.

LEE, K.; LEE, D.; KO, Y. W.; et al. An Objectified Naming System for Providing Context Transparent to Context-Aware Applications. 4th. Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, 2006.

LEUTNANT, V.; SCHMALENSTROEER, J.; POORTINGA, R. IST Amigo Project. Context Management Service Tutorial. Out. 2007. Disponível em: <https://gforge.inria.fr/frs/download.php/3229/CMS-Tutorial.pdf>. Acesso em: jun. 2011.

MCCARTHY, P. Introduction to Jena. 2004. Disponível em: <http://www.ibm.com/developerworks/xml/library/j-jena/>. Acesso em: maio 2011.

MEDLAR. Disponível em: <http://www.medlar.com.br>. Acesso em: maio 2011.

NOY, N. F.; FERGERSON, R. W.; MUSEN, M. A. The knowledge model of protege-2000: Combining interoperability and flexibility. In Proceedings of the 2nd International Conference on Knowledge Engineering and Knowledge Management (EKAW 2000), Juan-les-Pins, France, 2000.

OMG. CORBA. Disponível em: <http://www.corba.org/>. Acesso em: jun. 2011.

ORACLE1. Reflection. Disponível em: <http://download.oracle.com/javase/6/docs/technotes/guides/reflection/index.html>. Acesso em: junho de 2011.

ORACLE2. Interface Iterator. Disponível em: <http://download.oracle.com/javase/1.4.2/docs/api/java/util/Iterator.html>. Acesso em: jun. 2011.

ORACLE3. Netbeans. Disponível em: <http://netbeans.org>. Acesso em: jun. 2011.

ORACLE4. Creating a GUI With JFC/Swing. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/index.html>. Acesso em: jun. 2011.

PANDIS, I.; SOLDATOS, J.; PAAR, A.; REUTER, J.; CARRAS, M.; POLYMENAKOS, L. An ontology-based framework for dynamic resource management in ubiquitous computing environments. 2nd International Conference on Embedded Software and Systems, Northwestern Polytechnical University of Xi’an, P.R. China, 16–18. Dez. 2005.

PORTAL DO ESTUDANTE DE FILOSOFIA. Ontologia. Disponível em: <http://estudantedefilosofia.com.br/conceitos/ontologia.php>. Acesso em: maio 2011.

PAWAR, P.; TOMAKOFF, A. Ontology-based Context-Aware Service Discovery for Pervasive Environments. Proceedings of the First IEEE International Workshop on Service Integration in Pervasive Environments, 2006.

POWER, R.; LEWIS, D,; O'SULLIVAN, D.; CONLAN, O.; WADE, V. A Context Information Service using Ontology-Based Queries. Workshop on Advanced Context Modelling, Reasoning and Management, Sixth International Conference on Ubiquitous Computing, Nottingham, England, 7-10. Set. 2004.

PROCARE. Disponível em: <http://www.procaresaude.com.br>. Acesso em: maio 2011.

PRONEP. Disponível em: <http://www.pronep.com.br>. Acesso em: maio 2011.

RAMPARANY, F. et al. Context Management and Semantic Modelling for Ambient Intelligence. Technical Report TR-CTIT-06-52, Centre for Telematics and Information Technology University of Twente, Enschede. Ago. 2006. Disponível em: <http://130.89.10.12/~broens/dloadcontent/ramparany_FRCSS06.pdf>. Acesso em: jun. 2011.

RANGANATHAN, A.; CAMPBELL, R. H. A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. In Proceedings of ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, Jun. 2003.

RODRIGUES, A. L. B. (2009a) Uma Infra-estrutura para Monitoramento de Sistemas Cientes de Contexto. [MSc Thesis]. Programa de Pós Graduação em Engenharia Eletrônica, UERJ, Rio de Janeiro, Brazil, 2009.

RODRIGUES, A. L. B.; BEZERRA, L. N., SZTAJNBERG, A.; LOQUES, O. (2009b) Self-adaptation of fault tolerance requirements using contracts. Proceedings of IEEE/IFIP 7º International Conferences on Embedded and Ubiquitous Computing (EUC-09), Vancouver, Canadá, ago. 2009.

RODRIGUES, A. L. B.; GOMES I. C.; BEZERRA L. N.; SZTAJNBERG A.; CARVALHO S. T.; COPETTI, A.; LOQUES, O. (2009c) Using Discovery and Monitoring Services to Support Context-Aware Remote Assisted Living Applications. In: Workshop on Ubiquitous Computing, Management, and Embedded Applications in Healthcare (UbiHealth-09), Vancouver, Canadá, ago. 2009.

SABER ELETRÔNICA. Aplicativo da Valemobi. Janeiro 2009, p. 9. Ano 44. N. 432. Disponível em: <http://www.sabereletronica.com.br/secoes/leitura_noticia/353>. Acesso em: jun. 2011.

SCHMALENSTROEER, J.; LEUTNANT, V. et al. Amigo Context Management Service with Applications in Ambient Communication Scenarios. Constructing Ambient Intelligence, AmI 2007 Workshops Darmstadt, Germany. Nov. 2007.

SCIADS. Sistema Computacional Inteligente de Assistência Domiciliar à Saúde. Disponível em: <http://sites.google.com/site/sistemadetelesaude/home>. Acesso em: jun. 2011.

SLATER, S. G.; NEANDER, L.; CAREY, G. Measuring Quality of Life Outcomes Through the Use of Home Telehealth: Using a Case Study Model in a Terminal Heart Failure Patient. Home Health Care Management & Practice, Jun. 2006, 18: 333-335

SONG, D. et al. Ontology Application in Software Component Registry to Achieve Semantic Interoperability. In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II - Volume 02, IEEE Computer Society, 2005.

SUNSPOT. Sun Microsystems. Sun SPOT – Small Programmable Object Technology. Disponível em: <http://www.sunspotword.com>. Acesso em: maio 2011.

SZTAJNBERG, A.; RODRIGUES, A. L. B.; BEZERRA, L. N.; LOQUES, O.; COPETTI, A.; CARVALHO, S. T. Applying context-aware techniques to design remote assisted living applications. Int. Journal of Functional Informatics and Personalised Medicine, 2, 4 (Dec. 2009):358–378.

TELCOMED1. A Medic4All Group Company. WristClinic™ – The all-in-one wireless remote medical monitoring revolution. Disponível em: <http://www.telcomed.ie/allinone.html>. Acesso em: maio 2011.

TELCOMED2. A Medic4All Group Company. aLsis – Distress Alarm Pendant. Disponível em: <http://www.telcomed.ie/alsis.html>. Acesso em: jun. 2011.

TOSIN, C. E.; LIMA, J. R. L. A Reflexive Infrastructure for Context-Aware Applications. ubicomm, p.102-107, 2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, 2009.

VALEMOBI. Disponível em: <http://www.valemobi.com.br/>. Acesso em: jun. 2011.

VISIONTEC. VT7200E Conversor Digital Terrestre. Disponível em: <http://visiontec.com.br/conversores/vt7200e/>. Acesso em: jun. 2011.

W3C1. OWL Web Ontology Language - Use Cases and Requirements. W3C Recommendation, February 2004. (“What is an ontology?”). Disponível em: <http://www.w3.org/TR/webont-req/#onto-def>. Acesso em: maio 2011.

W3C2. OWL Web Ontology Language Overview, W3C Recommendation, February 2004. Disponível em: <http://www.w3.org/TR/owl-features/>. Acesso em: maio 2011.

W3C3. World Wide Web Consortium. Disponível em: <http://www.w3.org/>. Acesso em: maio 2011.

W3C4. Namespaces in XML 1.0, W3C Recommendation, Dezembro 2009. Disponível em: <http://www.w3.org/TR/xml-names/>. Acesso em: maio 2011.

W3C5. OWL 2 Web Ontology Language Document Overview, W3C Recommendation, Outubro 2009. Disponível em: <http://www.w3.org/TR/owl2-overview/>. Acesso em: maio 2011.

W3C6. Resource Description Framework (RDF). Fevereiro 2004. Disponível em: <http://www.w3.org/RDF/>. Acesso em: jun. 2011.

WEBSTER. Merriam-Webster. Disponível em: <http://www.merriam-webster.com>. Acesso em: maio 2011.

WIKIPEDIA. Comparison of Android devices. Disponível em: <http://en.wikipedia.org/wiki/List_of_Android_devices> Acesso em: maio 2011.

XSTREAM. Package com.thoughtworks.xstream. Disponível em: <http://xstream.codehaus.org/javadoc/com/thoughtworks/xstream/package-summary.html>. Acesso em: jun. 2011.

YE, J.; COYLE, L.; DOBSON, S.; NIXON, P. Ontology-based models in pervasive computing systems. The Knowledge Engineering Review, 22:315-347, Dez. 2007.