UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

VINÍCIUS MARAN

Modelo de Consulta de Dados Relacionais Baseada em Contexto para Sistemas Ubíquos

Tese apresentada como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Prof. Dr. José Palazzo M. de Oliveira

Porto Alegre
2016

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Maran, Vinícius

Modelo de Consulta de Dados Relacionais Baseada em Contexto para Sistemas Ubíquos. – 2016.

170 f.: 70 il.

Orientador: José Palazzo M. de Oliveira.

Tese (Doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2016.

1.Sensibilidade ao contexto. 2.Ontologia 3.Consulta de Dados. I. José Palazzo M. de Oliveira. II. Modelo de Consulta de Dados de Domínio Baseada em Contexto para Sistemas Ubíquos.

AGRADECIMENTOS

Em primeiro lugar, agradeço à Deus, por tudo e por todos. Agradeço a meus pais, Paulino e Leda e à minha irmã Ana Paula pelo auxílio, apoio, paciência e compreensão durante o período de realização deste trabalho. Agradeço de forma sincera todos os dias por ter uma base sólida proporcionada pela minha família, que contribuiu diretamente para a conclusão desta etapa;

à minha família, pela compreensão nos momentos em que estive ausente e pelo apoio durante o período de realização do doutorado;

ao meu orientador, professor José Palazzo M. de Oliveira pelo conhecimento e experiência repassados durante este período de trabalho. Sinto-me honrado pela oportunidade de ter realizado o curso de doutorado na Universidade Federal do Rio Grande do Sul e orgulhoso por ter sido orientado pelo professor José Palazzo. Levarei para o resto da vida os conselhos, ensinamentos e o exemplo de um professor e pesquisador exemplar;

aos professores Iara Augustin e Ricardo Pietrobon, que sempre me auxiliaram no desenvolvimento do trabalho e sempre se demonstraram animados e dispostos para me auxiliar no que fosse preciso;

aos colegas do laboratório 213, pelas discussões, comentários e auxílio nas reuniões e encontros do grupo. Em especial agradeço aos colegas Alencar Machado, Guilherme Medeiros Machado e Jonas Gassen;

à UNIJUI e aos meus ex-colegas de trabalho pelo apoio e compreensão no período em que trabalhei na universidade;

à Universidade Federal de Santa Maria (campus Cachoeira do Sul), aos meus colegas de trabalho e amigos, Docentes e Técnico-Administrativos pelo auxílio durante o desenvolvimento do trabalho e por último, agradeço à Universidade Federal do Rio Grande do Sul pelo suporte prestado na realização do doutorado e as pessoas que contribuíram de forma direta ou indireta na realização deste trabalho.

RESUMO

A computação ubíqua define que a computação deve estar presente em ambientes para auxiliar o usuário na realização de suas tarefas diárias de forma eficiente. Para que isto aconteça, sistemas considerados ubíquos devem ser conhecedores do contexto e devem adaptar seu funcionamento em relação aos contextos capturados do ambiente. Informações de contexto podem ser representadas de diversas formas em sistemas computacionais e pesquisas recentes demonstram que a representação destas informações baseada em ontologias apresenta vantagens importantes se comparada à outras soluções, destacando-se principalmente o alto nível de expressividade e a padronização de linguagens para a representação de ontologias. Informações consideradas específicas de domínio são frequentemente representadas em bancos de dados relacionais. Esta diferença em relação a modelos de representação, com o uso de ontologias para representação de contexto e representação relacional para informações de domínio, implica em uma série de problemas no que se refere à adaptação e distribuição de conteúdo em arquiteturas ubíquas. Dentre os principais problemas pode-se destacar a dificuldade de alinhamento entre as informações de domínio e de contexto, a dificuldade na distribuição destas informações entre arquiteturas ubíquas e as diferenças entre modelagens de contexto e de domínio (o conhecimento sobre os objetos do domínio). Este trabalho apresenta um *framework* de consulta entre informações de contexto e informações de domínio. Com a aplicação deste *framework*, a recuperação contextualizada de informações se tornou possível, utilizando a expressividade necessária para a modelagem de contexto através de ontologias e utilizando esquemas relacionais previamente definidos e utilizados por sistemas de informação. Para realizar a avaliação do *framework*, o mesmo foi aplicado em um ambiente baseado no cenário motivador de pesquisa, que descreve possíveis situações de utilização de tecnologias ubíquas. Através da aplicação do *framework* no cenário motivador, foi possível verificar que a proposta foi capaz de realizar a integração entre contexto e domínio e permitiu estender a filtragem de consultas relacionais.

Palavras-chave: Sensibilidade ao contexto, Ontologia, Consulta de Dados, Computação Ubíqua, Banco de Dados.

Model of Relational Data Querying Based on Context Modelling for Ubiquitous Systems

ABSTRACT

Ubiquitous computing defines the computer must be present in environments to assist the user to perform their daily tasks efficiently. Thus, ubiquitous systems must be aware of the context and should adapt its operation in relation to the captured environment contexts. Context information can be represented in different ways in computer systems, and recent research shows that the representation of context in ontologies offers important advantages when compared to other solutions, in particular, the high level of expressiveness and the standardization of languages for representation of ontologies. Domain specific information is frequently maintained in relational databases. This difference of representation models, using ontologies for context representation and relational representation to domain information, involves a number of problems as the adjustment and distribution of content in ubiquitous architectures. Related problems include the difficulty of alignment between field and context information, the difficulty in the distribution of information between ubiquitous architectures, and differences between the context and domain modeling (knowledge about the domain objects). This PhD thesis presents a framework of query for context information and domain information. On applying this framework, contextualized information retrieval becomes possible using the expressiveness required for context modeling using ontologies, and using relational schemas previously defined and used by information systems. In order to evaluate the framework, it was applied in an environment based on the motivating scenario. It was possible to verify that the framework was able to accomplish the integration of context and domain, and allowed the extension of the filtering relational queries.

Keywords: Context-awareness, Ontology, Query, Ubiquitous computing, Information Systems, Database.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABox	*Assertion Box*
BDR	Banco de Dados Relacional
CC/PP	*Composite Capabilities / Preference Profile*
CML	*Context Modeling Language*
DSL	*Domain Specific Language*
EM	Entrevista Motivacional
JSON-LD	*JavaScript Object Notation for Linked Data*
MOOC	*Massive Open Online Course*
ORM	*Object-Role Modeling*
OWL	*Ontology Web Language*
OWL-DL	*Ontology Web Language – Description Logics*
OWL-S	*Semantic Markup for Web Services*
RDF	*Resource Description Framework*
SGBD	Sistema Gerenciador de Banco de Dados
SPARQL	*SPARQL Protocol and RDF Query Language*
SQL	*Structured Query Language*
SQWRL	*Semantic Query Web Rule Language*
SWRL	*Semantic Web Rule Language*
TBox	*Terminological Box*
UAProf	*User Agent Profile*
UML	*Unified Modeling Language*
XCML	*XML Serialization of CML*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Mark Weiser (1991) apresentou no artigo *The Computer for the 21st Century* um conjunto de cenários onde a computação se tornaria onipresente nos ambientes dos usuários, ou seja, a computação poderia ser utilizada de diversas maneiras por usuários na resolução de problemas cotidianos mesmo que os usuários não conseguissem perceber a computação envolvida na realização destas atividades.

A partir destas motivações surgiu o termo Computação Ubíqua, que define a presença constante e imperceptível da computação perante o usuário e o ambiente. Apesar de ser um tema relativamente antigo, ainda não existem casos de implementação plena da computação ubíqua completamente alinhadas com a visão de Weiser. Isto se deve principalmente ao fato de que tornar a computação onipresente envolve uma série de questões tecnológicas e culturais, ambas de alta complexidade.

Muitas tecnologias e metodologias foram criadas e aprimoradas com vistas à aplicação em desenvolvimento de sistemas considerados ubíquos. Dentre as principais tecnologias e metodologias, pode-se destacar: (i) RSSF (Redes de Sensores Sem Fio) (GAMA & GABER, 2007), Internet das Coisas (*Internet of Things* - IoT) (PERERA et al., 2014), Computação Móvel, Computação Pervasiva, Web Semântica (BERNERS-LEE et al., 2001), Sensibilidade ao Contexto e à Situações (MOSTEFAOUI et al., 2004).

A área de sensibilidade ao contexto é essencial para a computação ubíqua (DEY et al., 2001). Computação sensível ao contexto pode ser definida como um conjunto de aplicações ou arquiteturas que se adaptam e personalizam sua execução baseando-se em informações provenientes do ambiente da qual estão inseridas (DEY et al., 2001). Em arquiteturas ubíquas, a sensibilidade ao contexto pode interferir diretamente em diversas operações, dentre elas destacam-se (CHALMERS, 2002):

- **Abstração de Informações de Contexto do Ambiente**: Informações são coletadas do ambiente (geralmente por sensores) ou informadas explicitamente por usuários e sistemas e são agregadas para formar informações de contexto com abstrações de alto nível. Estas informações descrevem o estado atual do ambiente e podem ser apresentadas diretamente para o usuário ou utilizadas por sistemas ubíquos;

- **Associação de Contextos com Dados**: Dados de domínio utilizados em arquiteturas ubíquas podem receber anotações semânticas que incorporam informações de contexto, como por exemplo, informações de localização onde a informação foi gerada ou informações de controle de acesso a determinados dados;

- **Descoberta de Recursos Baseada em Contexto**: Recursos podem ser utilizados por sistemas de acordo com as informações de contexto sobre os mesmos. Por exemplo, se um sistema precisa utilizar o recurso *"Impressora"*, ele deve ser capaz de encontrar a impressora mais próxima, realizar a conexão e utilizar o recurso quando necessário;

- **Ações Baseadas em Contexto**: Ações podem ser tomadas por sistemas ubíquos com base em informações de contexto captadas do ambiente, ou seja, informações de contexto podem ser utilizadas como gatilho para a execução de uma determinada ação, previamente modelada no sistema;

- **Seleção de Serviços Baseada em Contexto**: Informações de contexto podem servir como mediadores na utilização de serviços. O contexto pode definir parâmetros, limites e descrever preferências de usuários que podem ser utilizadas por serviços.

A associação de informações de contexto com dados de domínio para apresentação personalizada de informações tem sido intensamente estudada na academia e na indústria. Esta área apresenta uma série de problemas que ainda não tem soluções aceitas de forma unânime.

1.1 Contextualização

Estudos (BETTINI et al., 2010) (MAKRIS et al., 2013) (PERERA et al., 2014) demonstram que a modelagem de contexto baseada em ontologias é a que melhor atende a uma série de requisitos importantes para sistemas ubíquos[1], como por exemplo, a extensibilidade e nível de formalidade necessários para representar contextos.

Ontologias podem ser representadas em diversas linguagens. Estas linguagens podem ser classificadas como: (a) linguagens baseadas em lógica e (b) linguagens serializadas em XML. O segundo grupo é o mais utilizado atualmente devido à existência de padrões de representação gerenciados pela W3C (SUÁREZ-FIGUEROA et al., 2011).

Desta forma, baseado na modelagem de contexto em ontologias, ferramentas e metodologias têm sido desenvolvidas para persistir e recuperar estas informações (ZHANG et al., 2013) (LEE et al., 2014) (SAMWALD et al., 2013) em sistemas ubíquos. Porém, nem todas as informações utilizadas em sistemas ubíquos são representadas em ontologias, principalmente no que se refere às informações específicas de domínio, que são frequentemente representadas de forma relacional e persistidas em bancos de dados

[1] Um estudo mais detalhado é apresentado na Seção 3.1.

relacionais (BOLCHINI et al., 2013). Isto se deve ao fato de que estes bancos de dados foram modelados para serem utilizados por sistemas de informação considerados não ubíquos (BOLCHINI et al., 2013). Sistemas ubíquos preveem a utilização de fontes de informação diversas, inclusive de bancos de dados utilizados por outros sistemas, para recuperar conteúdo relacionado ao usuário (FORTE et al., 2008).

Portanto, para que estes dados de domínio, persistidos em bancos de dados relacionais, possam ser recuperados de forma contextualizada por sistemas ubíquos faz-se necessária a criação de formas de interligação de informações de contexto, que são frequentemente representadas em ontologias, e informações de domínio, modeladas e persistidas em bancos de dados relacionais (ADOMAVICIUS et al., 2011) (BOLCHINI et al., 2013). Além disso, sistemas de informação podem acessar bancos de dados relacionais, o que dificulta o processo de integração de sensibilidade ao contexto na recuperação destes dados.

Trabalhos recentes[2] propuseram a definição de modelos, extensões de linguagens de consulta e *frameworks* para a integração entre informações de contexto e informações de domínio. Porém, nenhum destes trabalhos une a expressividade da representação de contexto em ontologias e a modelagem de domínio de forma relacional sem a necessidade de reescrita das consultas realizadas no banco de dados.

1.2 Questão de Pesquisa

A partir da discussão apresentada nas seções anteriores, a questão de pesquisa definida é:

> *De que forma informações de contexto relacionadas a ambientes ubíquos podem ser utilizadas para reconfigurar consultas em bancos de dados relacionais?*

Analisando a questão de pesquisa, observa-se que a mesma está relacionada diretamente a consulta de dados em bancos de dados relacionais utilizando informações de contexto no processo de filtragem da informação. Assim, esta questão de pesquisa está relacionada diretamente às duas características da área de consulta de dados:

- **Armazenamento**: *De que forma informações de contexto podem ser utilizadas de forma externa à modelagem conceitual do banco de dados?* Ou seja, persistidas em bancos de dados, mas sem a necessidade de recriação ou modificação nas tabelas previamente criadas no banco de dados e consultas previamente definidas;

[2] O comparativo entre os trabalhos é apresentado no Capítulo 4.

- **Consulta**: *De que forma informações de contexto, utilizadas por sistemas ubíquos, podem ser utilizadas como filtro na realização de consultas em bancos de dados relacionais?*

1.3 Hipóteses de Pesquisa

Ontologias são frequentemente utilizadas para representar informações de contexto em sistemas ubíquos (BETTINI et al., 2010) (STRANG; LINNHOFF-POPIEN, 2004) e apresentam uma série de vantagens em relação a outras formas de representação de contexto, como por exemplo a existência de padrões de representação e alto nível de expressividade. Desta forma, a hipótese de pesquisa definida neste trabalho é:

> *Um framework de consulta suportado por rede de ontologias pode ser utilizado para integrar informações de domínio e contexto para estender uma consulta relacional.*

Esta hipótese de pesquisa é dividida em duas sub-hipóteses, apresentadas a seguir:

- **Sub-Hipótese 1**: Informações de contexto modeladas em ontologias podem ser alinhadas com ontologias que descrevem um banco de dados de domínio e persistidas na mesma instância de banco de dados, evitando assim a necessidade de utilização de mais de um modelo de banco de dados para o armazenamento e recuperação destas informações;

- **Sub-Hipótese 2**: Extensões de uma linguagem de consulta em bancos de dados relacionais podem utilizar definições de ontologias que descrevem contextos para recuperar informações contextualizadas, sem a necessidade de reescrita da consulta de forma manual.

1.4 Objetivo Geral e Objetivos Específicos

Este trabalho tem como objetivo geral definir e avaliar um *framework* de consulta de informações específicas de domínio modelado de forma relacional baseado em informações de contexto modeladas em ontologias. Para fins de avaliação, propõe-se a definição de um protótipo de arquitetura de software que suporte o *framework*. Esta arquitetura é desenvolvida e implementada como um sistema gerenciador de persistência e recuperação de dados. Desta forma, sistemas ubíquos podem utilizar funcionalidades da arquitetura.

As etapas a serem cumpridas para atingir o objetivo geral são as seguintes:

- Definição de um modelo baseado em ontologias para representar entidades de contexto genéricas e permitir o alinhamento de ontologias de contexto e ontologias que descrevem informações específicas de domínio;

- Definição de um *framework* de integração entre informações de contexto e informações especificas de domínio;
- Definição de uma metodologia de aplicação do *framework*;
- Prototipação da arquitetura de software implementando os modelos propostos anteriormente, para fins de avaliação;
- Avaliação do *framework* proposto através de prototipação e realização de testes em um ambiente controlado com dados de contexto coletados do ambiente e um banco de dados com informações específicas de domínio.

1.5 Organização do Texto

Esta tese está estruturada da seguinte forma:

- o capítulo 2 apresenta o cenário motivador relacionado à área de pesquisa de consultas em bancos de dados baseadas em contexto. O cenário apresenta as características de ambientes considerados ubíquos e a integração de tecnologias ubíquas com sistemas de informação;

- o capítulo 3 apresenta os principais conceitos das áreas de sensibilidade ao contexto, computação ubíqua e sensibilidade ao contexto, com foco na modelagem de informações neste tipo de sistema;

- o capítulo 4 apresenta um estudo de trabalhos que representam o estado da arte na área de integração entre sensibilidade à situações e informações de domínio. Neste capítulo também é apresentado um estudo comparativo entre os trabalhos, utilizando uma série de critérios de avaliação;

- o capítulo 5 apresenta o *framework* de consulta de dados de domínio, modelados de forma relacional, para a realização de consultas contextualizadas;

- o capítulo 6 apresenta uma arquitetura de software que gerencia o *framework* apresentado nesta tese. A prototipação desta arquitetura foi utilizada no processo de avaliação do *framework*;

- o capítulo 7 apresenta o processo de avaliação do *framework* apresentado nesta tese em um cenário de aplicação, para verificar a viabilidade de utilização do *framework* em sua utilização prática;

- o capítulo 8 apresenta as conclusões deste trabalho, apresentando as contribuições da pesquisa frente a literatura atual, uma lista de possibilidades de trabalhos futuros relacionados

a pesquisa e uma lista de publicações realizadas durante o período de doutoramento e relacionados a pesquisa.

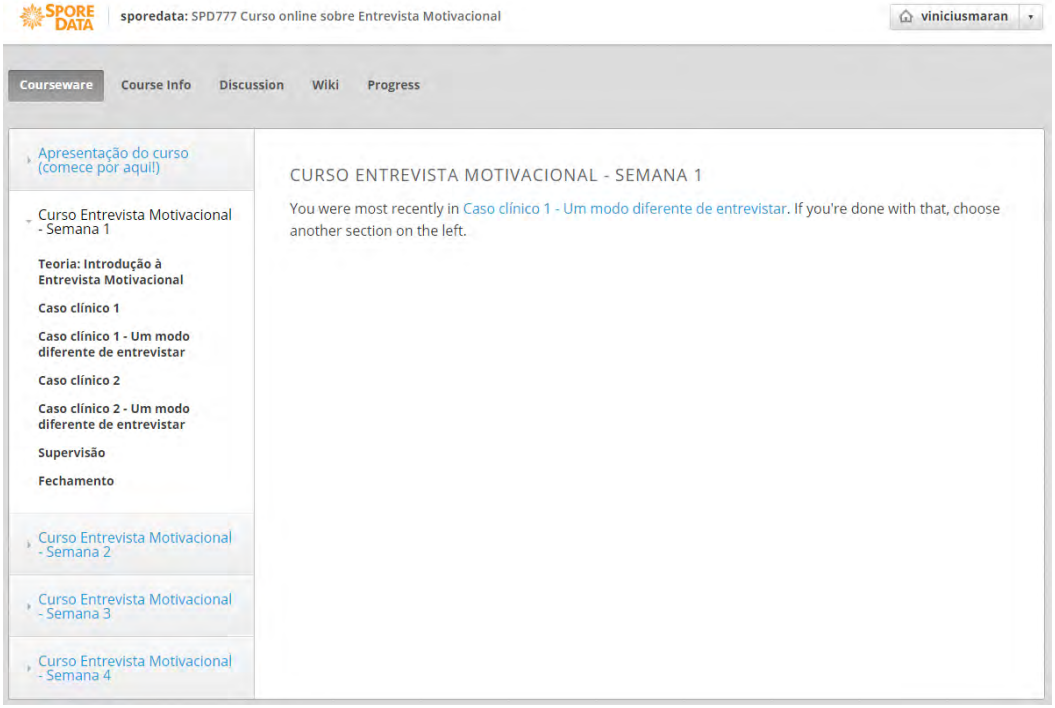
2 CENÁRIO MOTIVADOR

Neste capítulo é apresentado o cenário motivador da pesquisa realizada nesta tese de doutorado. O cenário motivador trata da aplicação de um *Massive Open Online Course* (MOOC) em um ambiente ubíquo.

2.1 Aplicação de um MOOC em um Ambiente Ubíquo

O OpenEDX (EDX, 2015) é uma plataforma *open source* de ensino a distância utilizada em diversas universidades e em cursos de várias áreas do conhecimento, como por exemplo, em cursos de computação no MIT ou direito em Harvard. Um exemplo de interface no ambiente OpenEDX em um curso de Entrevista Motivacional (EM) é apresentado na Figura 2.1.

Figura 2.1 – Exemplo de curso de Entrevista Motivacional na plataforma OpenEDX



Fonte: Autoria Própria.

EM é uma abordagem da área de psicologia centrada no paciente para demonstrar mudanças comportamentais para aumentar a efetividade no tratamento de doenças, como obesidade e tabagismo. O grupo de técnicas que compõem EM tem sido constantemente estudado em uma grande variação de comportamentos relacionados à saúde (COLE et al.,

2011). Entretanto, este grupo de técnicas ainda não é popular entre profissionais de saúde. Portanto, é extremamente importante a disseminação dos conceitos de EM para profissionais de saúde em geral (ANSTISS, 2009). EM possui um conjunto de princípios, entre eles, destacam-se (ARKOWITZ et al, 2015):

1) Resistir ao Reflexo de Correção: Profissionais de saúde que são iniciantes na profissão, muitas vezes têm um forte desejo de acertar as coisas, para curar, prevenir danos e promover o bem-estar do paciente. Ao ver o paciente seguir por um caminho errado, geralmente os profissionais tendem a querer sair na frente do paciente e dizer: *"Pare! Vire para trás! Há uma maneira melhor"*. Esta motivação, o desejo de corrigir o outro fica tão explícita nestes casos que se torna automática, quase reflexiva;

2) Entender as Motivações do Paciente: São as motivações dos pacientes, e não as dos profissionais de saúde, que tem maior potencial para motivar uma mudança de comportamento do paciente. Desta forma, um dos princípios em EM deve ser demonstrar interesse nos valores e motivações do paciente, para que seja possível explorar as percepções dos pacientes sobre suas situações atuais e suas próprias motivações para a mudança;

3) Animar o Paciente: Resultados de EM tendem a ser melhores quando pacientes se tornam mais ativos e aumentam o interesse na própria saúde. Desta forma, o papel do profissional de saúde é auxiliar os pacientes a explorar como eles podem fazer a diferença na sua própria saúde. Mais uma vez, idéias e recursos do próprio paciente são fundamentais nesta etapa.

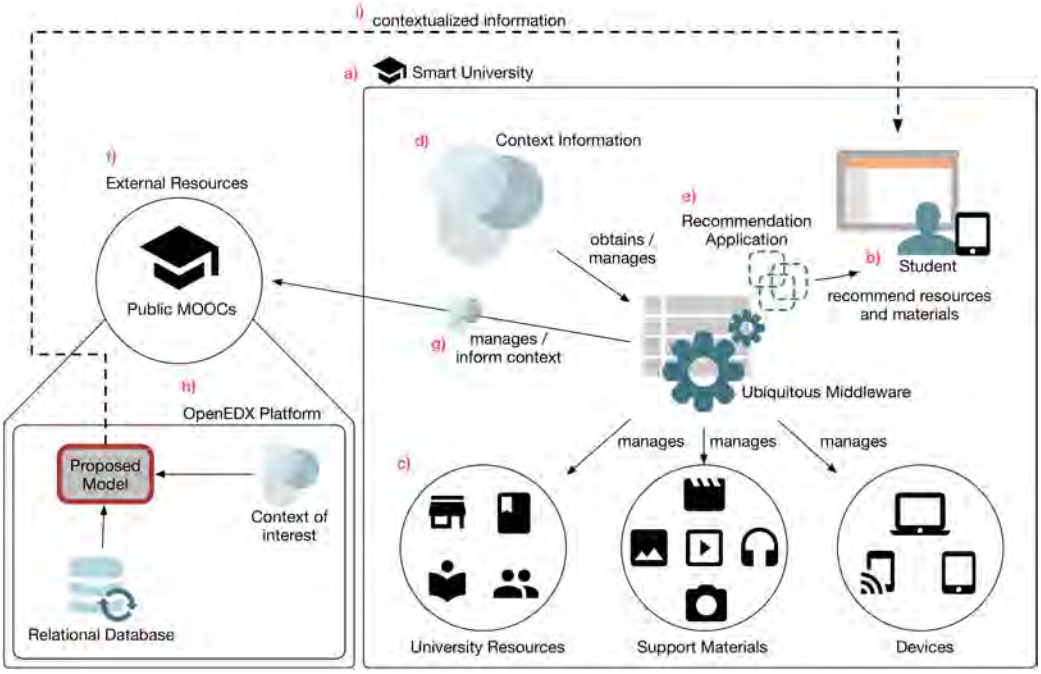
Desta forma, informações de contextos e situações de pacientes são essenciais para a aplicação de técnicas de EM em consultas. Assim, é necessário que existam formas para que MOOCs relacionados à área de EM possam apresentar materiais baseados em situações que possam ocorrer no cotidiano dos profissionais de saúde de acordo com informações relevantes de contexto do profissional de saúde.

Uma visão geral do cenário motivador apresentado neste capítulo é apresentada na Figura 2.2. Trabalhos recentes (STAVROPOULOS et al., 2010) (STAVROPOULOS et al., 2013) (BUENO-DELGADO et al., 2012) (MACHADO; PALAZZO, 2014) descrevem ambientes universitários inteligentes que possuem uma grande variedade de recursos educacionais que são gerenciados por sistemas inteligentes **(a)** e que podem ser recomendados aos alunos que estudam nestas universidades (KONGSAKUN; KUNG, 2012).

João é um estudante **(b)** no Curso de Saúde Coletiva em um campus universitário inteligente, que possui um sistema de recomendação **(e)** de conteúdos e recursos **(c)**.

Atualmente, João está no 6º semestre do curso de Graduação em Saúde Coletiva e está cursando a disciplina *PSI01003 – Psicologia Aplicada à Saúde*.

Figura 2.2 – Cenário Motivador da Pesquisa



Fonte: Autoria própria.

Middlewares Ubíquos oferecem abstrações para que aplicações utilizem recursos e tecnologias presentes em ambientes ubíquos. Eles são responsáveis pelo gerenciamento de informações de contexto que descrevem o ambiente no qual estão inseridos. Em uma determinada aula, o *middleware* de gerenciamento de contexto passa as seguintes informações de contexto ao sistema de recomendação **(d)**:

- Perfil do Aluno: João, estudante, 6º semestre, curso de Saúde Coletiva, brasileiro, 20 anos, mora com os pais;

- **Contexto Educacional**: Disciplina PSI01003, aula 10, conteúdo programático: Introdução à Psicologia e Saúde Coletiva, Experiência: Nível básico de atenção à saúde (estágio em posto de saúde);

- Localização: Brasil, Cidade de Porto Alegre, Campus Universitário, Sala de aula 65-201;

- Dispositivo utilizado pelo aluno: *Smartphone*;

- Situação: aula; inicio: 08:15.

A partir destas informações, o sistema de recomendação recomenda ao aluno o livro *Psicologia e Saúde Coletiva*, que está na biblioteca mais próxima da sala de aula que João está no momento da recomendação.

Atualmente, cursos online abertos (*Massive Open Online Courses* - MOOC) **(f)** são oferecidos como uma oportunidade de ampliar os conhecimentos dos alunos. Estes MOOC podem ser cadastrados no sistema de recomendação como recursos externos à universidade. Assim, mesmo externos à universidade, estes cursos podem ser recomendados como uma fonte alternativa a alunos que desejem melhorar seus conhecimentos em um determinado assunto.

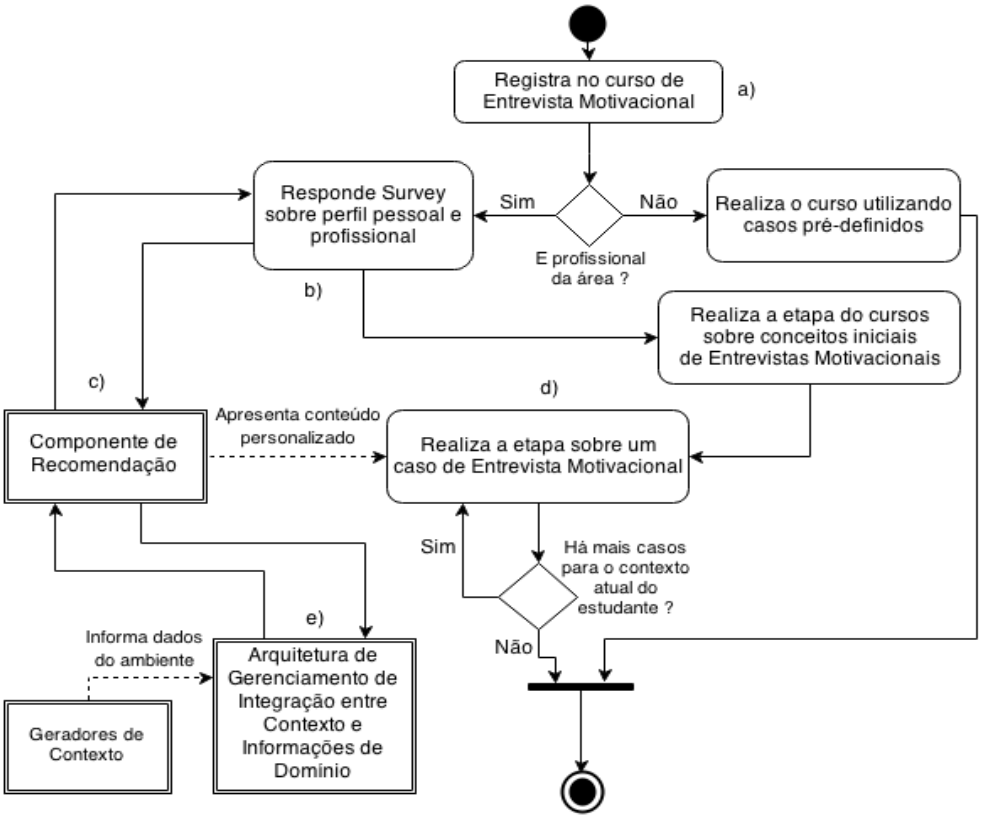
Em uma aula da disciplina *PSI01003 – Psicologia Aplicada à Saúde*, o professor apresentou o conceito de EM. No final da aula, o sistema de recomendação apresentou à João a recomendação de um MOOC oferecido por uma universidade brasileira sobre EM. O aluno, interessado em obter mais informações sobre o tema, se registra no MOOC, que é oferecido na plataforma *OpenEDX* **(h)**. No momento em que o aluno se registra no MOOC, o *middleware* ubíquo transfere informações de contexto (com a autorização do aluno) para o MOOC. Assim, informações de perfil e de dispositivos associados ao aluno podem ser utilizadas pelo MOOC na filtragem de informações do curso para o aluno. O MOOC sobre EM que João se inscreveu apresenta casos e questionamentos para avaliar o entendimento do aluno sobre as técnicas de entrevista motivacional utilizadas nos casos.

Neste cenário motivador, considera-se que a plataforma OpenEDX utiliza serviços de uma arquitetura externa. Esta por sua vez, realiza o gerenciamento da integração de informações específicas de domínio, relacionadas a cursos e alunos, e informações de contexto. Estas informações de contexto podem ser captadas do ambiente de forma dinâmica, ou estáticas, através de inferência sobre perfis e preferências de alunos, professores e cursos. A Figura 2.3 apresenta um diagrama de fluxo das atividades do aluno no curso de EM que utiliza a arquitetura.

No diagrama da figura, um aluno se registra no curso online de EM oferecido por uma universidade que utiliza a plataforma OpenEDX (a). No momento em que o aluno se cadastra no curso, ele é questionado sobre a sua profissão atual. Caso seja um profissional de saúde, um questionário é apresentado para que ele o responda (b). Este questionário é aplicado com uma série de questões que auxiliam no processo de escolha de materiais de apoio para este aluno, de acordo com o seu perfil e especialidade na qual trabalha. A especialidade na qual o profissional trabalha é de extrema importância, pois EM pode ser aplicada em diversas especialidades e de diferentes formas.

Após a conclusão do questionário e da etapa do curso que apresenta conceitos iniciais de EMs, o componente de recomendação, integrado à plataforma OpenEDX realiza uma requisição à arquitetura de gerenciamento de integração entre contexto e informações de domínio. Esta arquitetura gerencia o *framework* proposto nesta tese e responde a requisição com informações de domínio pertinentes ao contexto informado pelo componente de recomendação.

Figura 2.3 – Diagrama de fluxo das atividades básicas realizadas pelo aluno no cenário motivador.



Fonte: Autoria Própria.

O componente de recomendação então apresenta as informações em forma de casos de entrevistas motivacionais para o estudante. Caso uma nova situação de interesse do estudante seja inferida pela arquitetura de gerenciamento de integração entre contexto e informações de domínio, uma requisição é feita (e) ao componente de recomendação e este informa ao usuário a existência de um novo caso ou material importante para a realização do curso. Estes processos são realizados enquanto ocorrerem situações de interesse ou houver casos de interesse do profissional de saúde.

No contexto da aplicação de um curso de EM na plataforma OpenEDX com recomendação de materiais e casos para profissionais baseados em contexto, uma camada de

sensibilidade ao contexto deve ser integrada ao sistema, pois a plataforma OpenEDX não suporta nativamente esta funcionalidade. Além disso, a integração de sensibilidade ao contexto neste tipo de aplicação é uma tendência de acordo com estudos recentes na área (DILLENBOURG et al., 2014) (GUTIÍRREZ-ROJAS et al., 2014).

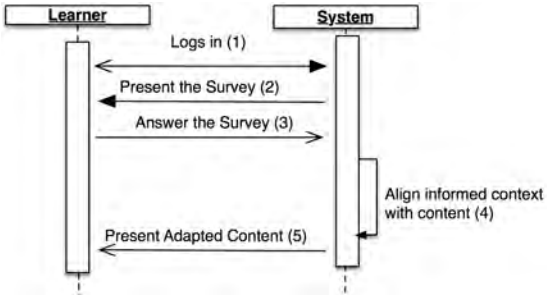
As principais ações a serem considerados neste estudo de caso são: (a) instrutor reúne informações sobre conceitos e situações em cada um de um grupo de alunos; (b) cada aluno recebe conceitos de interesse para ela, bem como situado dentro de seu próprio contexto com OpenEDX. Com base nesta ação simplificada, um fluxo de trabalho correspondente no OpenEDX seria aproximadamente da seguinte forma (Figura 2.4). Primeiro, os alunos se registram no sistema (1). Logados, eles recebem uma pesquisa (2) e a respondem (3) ao fazer uma seleção entre as seguintes opções:

(i) Quais são os dois conceitos que gostaria de aprender: (a) resistindo ao reflexo de endireitamento, (b) a ouvir a motivação do seu paciente, (c) que habilita o seu paciente;

(ii) Em que situação única que o aluno seria mais provável para aplicar cada um dos seguintes conceitos: (a) O exercício físico e (b) Cessação do Tabagismo.

Com base nas respostas, o sistema classifica o conteúdo disponível (4), apresentando o conteúdo (vídeos, exercícios e leituras) para os alunos (5).

Figura 2.4 – Diagrama de sequência de primeira interação do aluno com o curso.

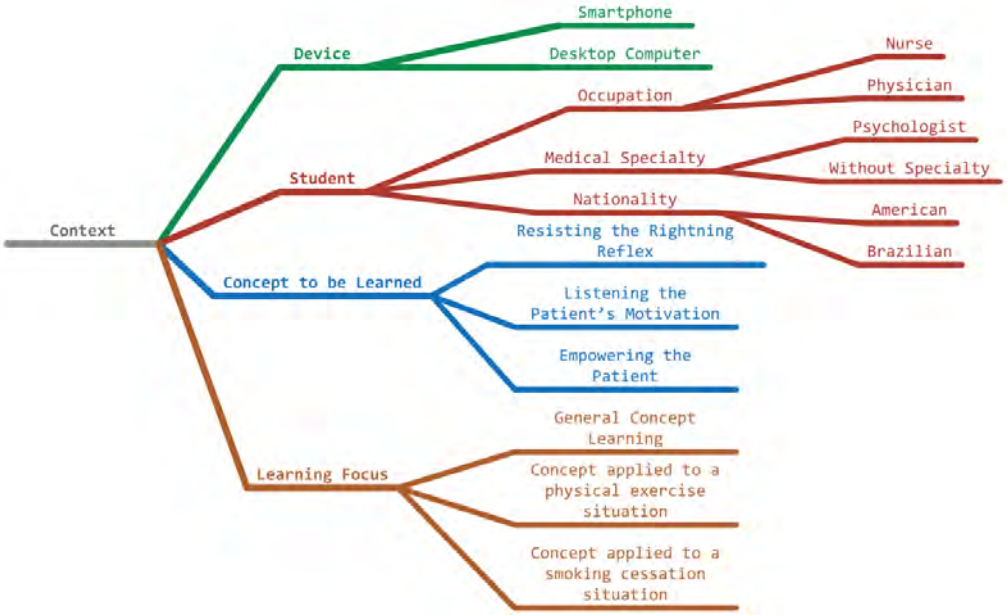


Fonte: Autoria Própria.

Por exemplo, se o aluno escolhe para aprender (a) o conceito de resistir ao reflexo de endireitamento em uma situação em que um paciente precisa melhorar seus hábitos de exercício físico e (b) o conceito de capacitar o seu paciente em uma situação em que o paciente gostaria para parar de fumar, então um modelo de consulta pode determinar que: (a) conceitos gerais (vídeo, exercícios e leituras) seriam entregues ao aluno; e (b) conceitos aplicados a suas respectivas situações (vídeo, exercícios e leituras) seriam entregues aos alunos.

Além das informações de contexto apresentadas anteriormente, cada material de apoio aos alunos e caso de EM pode estar associado a uma ou mais dimensões de contexto, para que assim possa ser recomendado ao aluno. No contexto deste cenário motivador, as dimensões de contexto que podem ser consideradas para associação com os materiais de apoio são apresentadas na Figura 2.5.

Figura 2.5 – Dimensões de contexto que podem ser associadas a materiais de apoio e a casos de EM no contexto do cenário motivador.

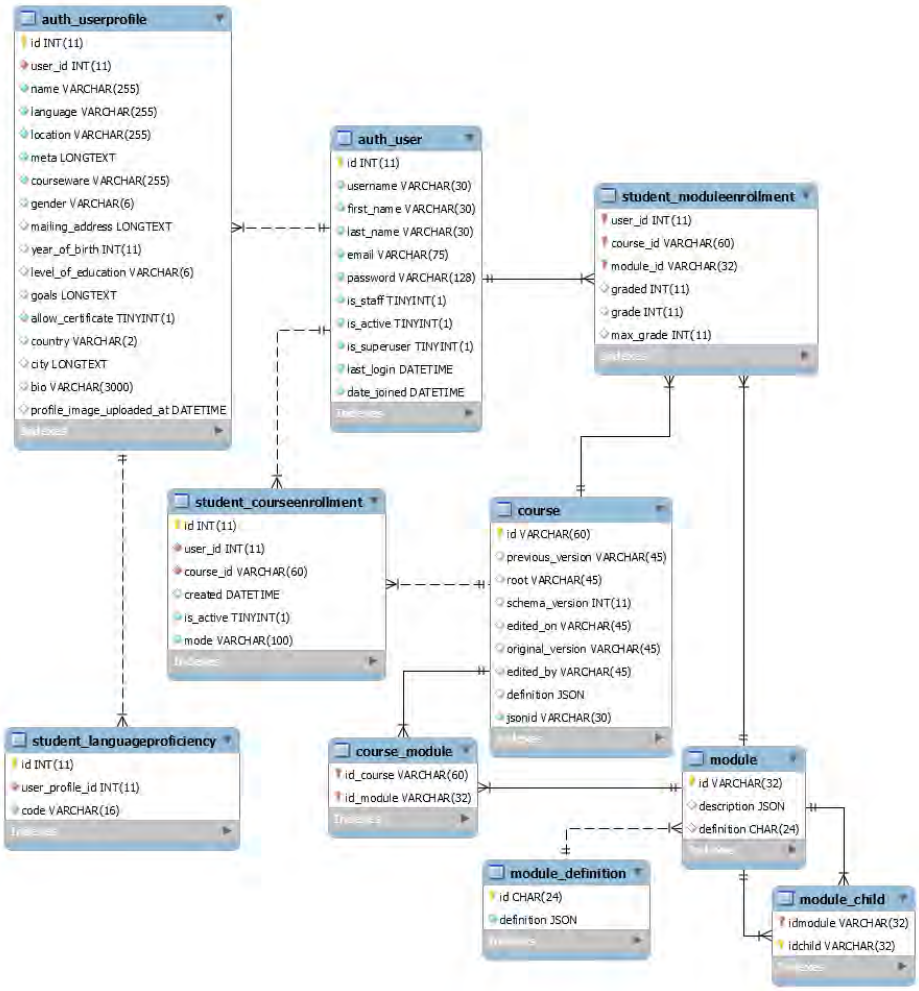


Fonte: Autoria Própria.

A plataforma OpenEDX utiliza um banco de dados relacional que armazena o conteúdo utilizado nos cursos e dados estatísticos de acesso. A estrutura das principais tabelas utilizadas para a consulta de casos de EM em um MOOC na plataforma OpenEDX é apresentada na Figura 2.6. O esquema lógico do banco de dados relacional utilizado na plataforma OpenEDX é apresentado no Apêndice A.

Considerando as dimensões de contexto apresentadas anteriormente, contextualizar as consultas realizadas no banco de dados utilizado pela plataforma OpenEDX através do uso de ontologias exigiria uma remodelação da plataforma e do esquema conceitual do banco de dados. Este processo de remodelação seria necessário porque as dimensões de contexto que são relevantes para a adaptação de conteúdo não foram contempladas no processo de modelagem do banco de dados relacional.

Figura 2.6 – Esquema de tabelas utilizadas na consulta de informações sobre materiais na plataforma OpenEDX.



Fonte: Autoria Própria.

2.2 Considerações Sobre o Cenário Motivador

Pode-se observar que existem características no cenário motivador que se assemelham a cenários apresentados em trabalhos da área de pesquisa (BOLCHINI et al., 2009) (FENG et al., 2004) (PERERA et al., 2013), apesar da diferença nos domínios de aplicação. Em relação à modelagem de contexto, observa-se que o contexto é modelado utilizando ontologias e utilizado por arquiteturas ubíquas para a escolha de serviços e recomendação de conteúdo.

Há uma troca de informações de contexto entre o *middleware* que gerencia as informações de contexto e sistemas externos. Esta característica é um problema de pesquisa abordado em trabalhos recentes (PERERA et al., 2014) (MAKRIS et al., 2013).

Além disso, existe um *gap* em relação a ligação entre informações de contexto e dados de domínio no que se refere a persistência e recuperação destes dados utilizando informações

de contexto no processo de consulta. Este *gap* está diretamente relacionado as duas características: **(i)** Informações relacionadas ao domínio de aplicação geralmente são modeladas de forma relacional e persistidas em bancos de dados relacionais. Isto é evidenciado pela grande adoção de gerenciadores de bancos de dados relacionais (BDENGINES, 2015), e **(ii)** Informações relacionadas a contexto, utilizadas por sistemas ubíquos sensíveis ao contexto são frequentemente modeladas em ontologias (BETTINI et al., 2010) (STRANG; LINNHOFF-POPIEN, 2004).

3 FUNDAMENTAÇÃO CONCEITUAL

Neste capítulo são apresentados os principais conceitos relacionados ao trabalho apresentado nesta tese. Inicialmente, na Seção 3.1, são apresentados os principais conceitos das áreas de computação ubíqua e computação sensível ao contexto. Posteriormente, na Seção 3.2, são apresentados os principais conceitos relacionados a área de sensibilidade à situação. Na Seção 3.3 são apresentados os conceitos relacionados à recuperação de dados estruturados de forma relacional baseada em contexto, área de pesquisa que está diretamente relacionada com a tese. Por fim, na Seção 3.4, são apresentadas as considerações sobre as áreas de pesquisa relacionadas à tese.

3.1 Sensibilidade ao Contexto

A área de *context-aware computing*, ou computação sensível ao contexto, está diretamente relacionada à área de computação ubíqua e tem evoluído constantemente em diversos aspectos. Nesta seção são apresentadas as principais características e trabalhos sobre definição, representação, persistência e recuperação de contexto.

3.1.1 Definição de Contexto

Em sistemas ubíquos, dados que descrevem o ambiente, coletados por sensores, são abstraídos em conjuntos e então chamados de informações de contexto. Algumas pesquisas buscaram definir o que é contexto. A primeira definição de contexto na computação foi apresentada por Schilit & Theimer (1994), onde contexto foi definido como localização, identificação de pessoas e objetos, e mudanças de estado de pessoas e objetos. Esta definição é difícil de ser aplicada de forma prática, pois não explica de forma clara quais informações podem ou não podem ser consideradas como informações de contexto.

Dey et al., (2001) apresentou uma nova definição de contexto e algumas características comuns em sistemas sensíveis ao contexto. Neste trabalho, contexto foi definido como *“Qualquer informação que possa ser utilizada para caracterizar a situação de entidades (pessoa, lugar ou objeto) que sejam consideradas relevantes para interação entre um usuário e uma aplicação (incluindo o usuário e a aplicação)”* (Tradução livre).

Contexto também pode ser definido com base em duas definições principais: (*i*) o contexto atua como um conjunto de restrições que influenciam o comportamento de um

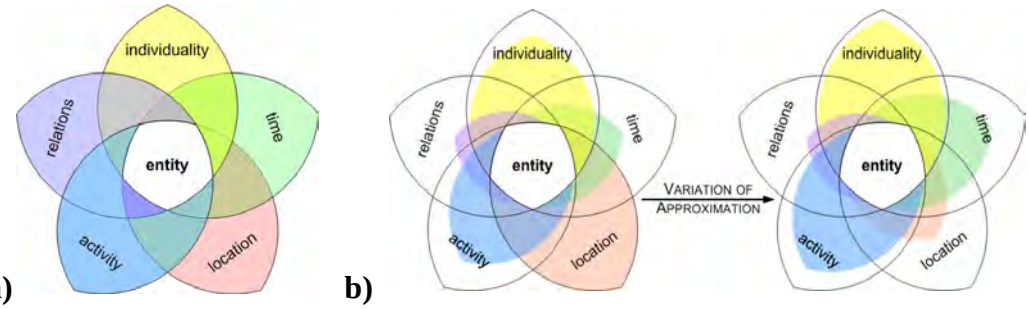
sistema, embutido em uma dada tarefa e (*ii*) a definição de contexto depende da área de conhecimento à qual pertence (BRÉZILLON; ARAÚJO, 2005).

Zimmermann et al., (2007) definiram uma forma operacional de contexto, separando a definição original em subcategorias de contexto (Figura 3.1a), denominadas: (*i*) *individuality* (propriedades e atributos que definem as entidades em si); (*ii*) *activity* (todas as tarefas que as entidades podem estar envolvidas); (*iii*) *location* e *time* (coordenadas espaço-temporais das entidades); e, (*iv*) *relations* (informações sobre qualquer relação que a entidade possa estabelecer com outras entidades).

Além da definição de contexto de uma forma operacional, Zimmermann et al., (2007) apresentou as formas de como o contexto pode variar de acordo com o tempo. De acordo com o trabalho, informações de contexto podem variar das seguintes formas:

(a) Variação de Aproximação: No processo de mudança de contextos em sistemas computacionais, as informações são coletadas e agregadas. Na agregação de informações, pode-se observar a ocorrência de especializações ou abstrações de dados, representando assim um contexto. Estas especializações ou abstrações ocorrem a partir de aproximações de valores. Desta forma, há constantes variações em relação à aproximação dos valores de elementos que juntos, representam o contexto. A Figura 3.1b apresenta um exemplo de variação da aproximação de valores, onde há variações nos valores de individualidade, localização, tempo e atividade;

Figura 3.1 – a) Categorias Fundamentais de Contexto; b) Variação de Aproximação.



a) **b)**

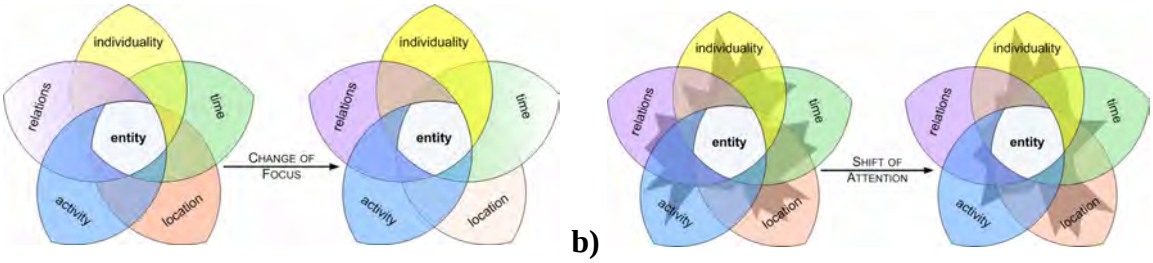
Fonte: (ZIMMERMANN et al., 2007).

(b) Mudança de Foco: O foco em relação ao contexto é modificado constantemente. Para uma entidade, a distância de localização e tempo da fonte de contexto (geralmente um sensor) determina se o elemento de contexto está em foco ou não (Figura 3.2a);

(c) Troca de Atenção: O foco da aplicação ou do sistema pode variar conforme o tempo. Por exemplo, em uma situação onde o sistema precisa analisar informações de

localização do usuário, a atenção da aplicação está na localização e no usuário. Em outra situação, como por exemplo, a análise das atividades realizadas por um determinado usuário, a atenção deixa de ser a localização e passa a ser o conjunto de atividades feitas pelo usuário. A representação destas situações é apresentada na Figura 3.2b.

Figura 3.2 – a) Mudança de Foco; b) Troca de Atenção.



a) **b)**

Fonte: (ZIMMERMANN et al., 2007).

As definições de Dey et al., (2001), Zimmermann et al., (2007) e Brézillon & Araújo (2005) são as mais utilizadas em trabalhos que utilizam computação sensível ao contexto. Com o passar do tempo e a evolução das arquiteturas baseadas em contexto, o conceito de contexto e a forma de utilização do mesmo foram modificadas. De acordo com estudos recentes (MAKRIS et al., 2013) (PERERA et al., 2014), há claras distinções entre as primeiras definições de contexto (definidas nos anos 90 e 2000) e a forma como o contexto é utilizado atualmente em sistemas computacionais.

A primeira diferença é que atualmente contexto é tratado como um conjunto de informações medidas e conhecimento inferido sobre entidades, ao invés de ser tratado como um conjunto de informações que definem o estado de uma entidade (STRASSNER; O'SULLIVAN, 2009). Ou seja, enquanto as primeiras definições de contexto apontavam que os dados que compõem contextos já podem ser chamados de contexto, como por exemplo, um dado de medição de temperatura. Nas mais recentes definições de contexto uma informação só pode ser considerada como contexto após ser gerada por uma inferência ou medição, como por exemplo, uma série de dados de temperatura lidos por um sistema ubíquo (números que representam a temperatura do ambiente) são considerados como informação de contexto após a realização de inferência sobre estes dados, onde por exemplo a informação *temperatura_quente* é inferida. Além disso, há uma falta de separação clara entre os conceitos de contexto e as informações de contexto (HENRICKSEN, 2003).

Trabalhos (STERRITT et al., 2005) (MAKRIS et al., 2013) (PERERA et al., 2014) afirmam que o contexto também pode surgir a partir da atividade geral de um sistema sensível

ao contexto, gerando e sustentando um determinado contexto. Contexto existe e varia mesmo sem que existam interações entre o usuário e o sistema (MAKRIS et al., 2013). Isto ocorre porque atualmente, especialmente em sistemas de rede sem fio, a falta de comunicação entre sistemas por um determinado tempo pode resultar na inferência de novos contextos. A Tabela 3.1 apresenta um comparativo relacionado as definições de contexto dos principais autores da área.

Tabela 3.1 – Características das definições de contexto propostas por alguns autores

Definições Clássicas (DEY et al., 2001), (ZIMMERMANN et al., 2007)	**Definições Recentes (MAKRIS et al., 2013), (STRASSNER; O'SULLIVAN, 2009)**
Contexto é um conjunto de valores numéricos	Contexto é um conhecimento medido e inferido
Contexto é o estado da informação	Contexto é um fluxo de informação
Contexto caracteriza a situação de uma entidade	Contexto surge das atividades gerais de um sistema sensível ao contexto
Contexto é resultante exclusivamente de interações	Contexto pode existir independentemente de interações
Os usuários participam dos processos de adaptação dos sistemas	Adaptações de sistemas não são perceptíveis aos usuários

Fonte: Tradução livre da tabela publicada por (MAKRIS et al., 2013).

Informações de contexto podem ser representadas de diversas formas em sistemas computacionais. A próxima seção apresenta as principais formas de representação de contexto e trabalhos relacionados à modelagem de contexto em sistemas ubíquos.

3.1.2 Modelagem de Contexto

Em paralelo à evolução das definições de contexto e do aumento da utilização da sensibilidade ao contexto em sistemas, a representação de contextos tem evoluído constantemente. Informações de contexto podem ser representadas e inferidas de diversas formas. As principais formas de modelagem de contexto são: (i) Modelo chave-valor, (ii) Esquemas de marcação, (iii) Modelos orientados a objetos, (iv) Modelos gráficos, (v) Modelos baseados em lógica e (vi) Modelos baseados em ontologias.

3.1.2.1 Modelo Chave-Valor

O modelo chave-valor consiste de conjuntos de pares com chave-valor, onde a busca é sempre realizada sobre a chave. O modelo de chave-valor foi utilizado nos primeiros

trabalhos de representação de contexto (SCHILIT; THEIMER, 1994). CONTORY (RIVA, 2006) é um *framework* para a utilização de uma modelagem de contexto baseada em triplas de chave-valor, combinada com a linguagem SQL para a consulta de contextos para dispositivos móveis. A seguir, é apresentado um exemplo de contexto representado em chave-valor da atividade *"caminhando em ambiente externo"*, com as variáveis barulho (*noise*), iluminação (*light*) e atividade (*activity*).

```
<noise = medium, light = natural, activity = walking>
```

Por ser simples de ser representado, o modelo chave-valor também é de fácil gerenciamento e é eficiente (em relação à velocidade de consulta). Porém, estes modelos apresentam sérios problemas para representar estruturas complexas de dados e contextos (BETTINI et al., 2010) (STRANG; LINNHOFF-POPIEN, 2004) (PERERA et al., 2014).

3.1.2.2 Esquemas de Marcação

Esquemas de marcação são estruturas hierárquicas que são marcadas com *tags* que definem a informação sobre a estrutura do dado. Geralmente, são derivados da *Standard Generic Markup Language* (SGML), como por exemplo o XML, e são representados na forma de perfis (*profiles*). Existem ainda extensões de padrões, como o CC/PP (*Composite Capabilities / Preference Profile*) (BUTLER; DELI, 2001) e UAProf (*User Agent Profile*) (ALLIANCE, 2003). Outros trabalhos propuseram extensões da linguagem XML para a representação de contextos. Rukzio et al., (2004) implementaram um esquema em XML para a representação simplificada de informações sobre usuários, dispositivos, serviços e redes para dispositivos móveis utilizando como base o padrão *3GPP Generic User Profile*.

A ContextML (KNAPPMEYER et al., 2010) foi definida como uma extensão da linguagem XML para a representação de contextos e passível de utilização com REST *webservices*.

A linguagem ContextML é baseada nas definições de *entity* (entidade – que representa toda informação que muda), e *scope* (escopo – onde uma entidade pode estar associada a diversos escopos, que por sua vez são separados por tipo, como localização, tempo, perfil, entre outros). A Figura 3.3 apresenta um exemplo de representação de um provedor de contexto (localização de um usuário) na linguagem ContextML. A XCML (*XML Serialization of CML*) é uma linguagem de serialização da linguagem CML (*Context Modeling Language*)

em XML. A conversão da linguagem CML para RDF ou OWL gera perdas semânticas e limita a inferência (ROBINSON et al., 2007).

Figura 3.3 – Exemplo de representação de contexto utilizando ContextML

```
<contextML><ctxEls><ctxEl>  <contextProvider id="PP" v="1.0.0"/>
  <entity id="john" type="username"/><requestEntity id="john" type="username"/>
  <scope>position</scope>
  <timestamp>2010-02-08T16:21:20+01:00</timestamp>
  <expires>2010-02-08T16:26:20+01:00</expires>
  <dataPart>
    <par n="latitude">52.281571</par> <par n="longitude">8.024918</par>
    <par n="accuracy">150.0</par>
  </dataPart>
</ctxEl></ctxEls></contextML>
```

Fonte: (KNAPPMEYER et al., 2010).

A CD-XML (*Context-Definition XML*) é uma extensão da linguagem XML para a definição de contextos e atualmente é utilizada na arquitetura MultiS (FEHLBERG et al., 2013). Esta linguagem foi desenvolvida para representar mudanças de contexto que devem modificar a forma de funcionamento da arquitetura MultiS. A Figura 3.4 apresenta um exemplo do contexto *"está em um cômodo"* representado na linguagem CD-XML.

Figura 3.4 – Exemplo de representação de contexto utilizando CD-XML

```
<?xml version="1.0" enconding="UTF-8"?>
<context label="WhenInRoom"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    xsi:noNamespaceSchemaLocation="cd-xml.xsd">
    <filter name="alert">
        <parameters>
            <parameter>present</parameter>
        </parameters>
    </filter>
    <sensor>currentTime</sensor>
    <context label="isPersonInRoom">
        <filter name="FilterAtLeast" default="false">
            <parameters>
                <parameter>3</parameter>
            </parameters>
        </filter>
        <sensor>isComputerInUse</sensor>
        <sensor>isChairInUse</sensor>
        <sensor>isMovActive</sensor>
        <sensor>isLightOn</sensor>
        <sensor>isDoorLocked</sensor>
    </context>
</context>
```

Fonte: (FEHLBERG et al., 2013).

Esquemas de marcação têm sido utilizados em muitos sistemas ubíquos para a transmissão de informações e armazenamento temporário de dados. Este modelo pode

representar estruturas mais complexas e flexíveis se comparado à modelagem utilizando chave-valor. Porém, depende diretamente da aplicação, pois não há padrões consolidados para a representação da estrutura de contextos. Para a descrição de sensores, há o padrão SensorML (ROBIN; BOTTS, 2006). Além disso, dependendo da complexidade dos relacionamentos entre informações estruturadas em esquemas distintos, a recuperação da informação pode se tornar difícil e complexa.

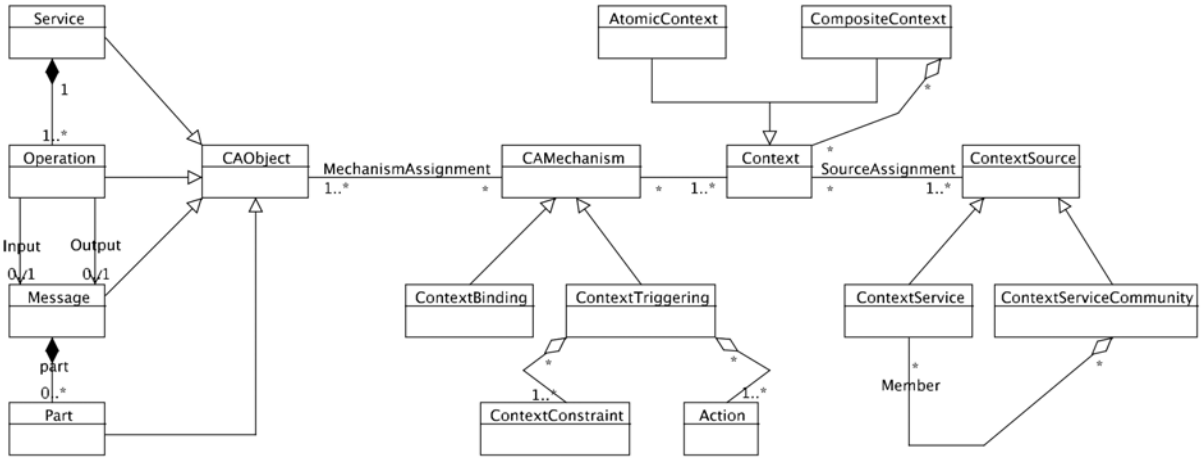
3.1.2.3 Modelos Gráficos

Modelos gráficos são utilizados para representar de forma gráfica os relacionamentos entre as entidades que representam contexto. As linguagens mais utilizadas para modelagem gráfica são: UML – *Unified Modeling Language*, ORM – *Object Role Modeling* (HENRICKSEN; INDULSKA, 2006), e CML – *Context Modeling Language* (HENRICKSEN et al., 2005).

A CML é uma proposta de extensão da ORM para a representação de contextos de forma gráfica. Ela define um conjunto de extensões para atender requisitos comuns na modelagem de contextos. Estes requisitos são: (i) Representação de fatos estáticos e dinâmicos, (ii) Representação de fatos alternativos, (iii) Representação de fatos temporais e (iv) Representação de qualidade da informação (HENRICKSEN et al., 2005).

A ContextUML (SHENG et al., 2005) foi proposta como uma extensão da linguagem UML para a apresentação de serviços para sistemas sensíveis ao contexto. Basicamente, ela define um modelo com classes pré-definidas para a representação de informações importantes em sistemas ubíquos. O metamodelo proposto na ContextUML é apresentado na Figura 3.5.

Figura 3.5 – Metamodelo da linguagem ContextUML



Fonte: (SHENG et al., 2005).

Além de extensões da linguagem UML, trabalhos têm sido propostos para a representação gráfica de contextos utilizando DSL (*Domain Specific Languages*). A MLContext (HOYOS et al., 2010) propôs uma definição de alta abstração e independente de plataforma para a representação de contextos.

Representações gráficas são de fácil entendimento e facilitam o processo de desenvolvimento de sistemas ubíquos. Além disso, tem como principal característica a possibilidade de utilização de diversas abordagens para a representação de nível mais baixo dos diagramas (STRANG; LINNHOFF-POPIEN, 2004) (PERERA et al., 2014).

Uma representação de baixo nível de um diagrama pode ser, por exemplo, um documento XML ou um banco de dados relacional. Esta característica permite a flexibilidade na implementação destes modelos, porém dependendo da modelagem realizada, a interoperabilidade entre modelos ou a inferência sobre informações modeladas se torna complexa (STRANG; LINNHOFF-POPIEN, 2004) (PERERA et al., 2014).

3.1.2.4 Modelos Orientados a Objeto

Modelos de contexto orientados a objetos são modelos baseados em hierarquias de classes, com relacionamentos, que seguem as diretrizes do paradigma de programação orientado a objetos.

Desta forma, estes modelos promovem a reutilização de modelagens e encapsulamento da informação (STRANG; LINNHOFF-POPIEN, 2004) (PERERA et al., 2014). Como a maioria das linguagens de programação utiliza o paradigma orientado a objetos, modelos de contexto baseados em orientação a objetos tendem a ser fortemente acoplados a linguagens de programação e ser facilmente aplicáveis a sistemas ubíquos. Este fato dificulta a distribuição de informações de contexto entre módulos de sistemas e entre diferentes linguagens de programação (característica comum em sistemas ubíquos).

Modelos Orientados a Objetos podem ser implementados de duas formas: (i) diretamente na programação de sistemas ubíquos com a utilização de linguagens orientadas a objetos, e (ii) através da utilização de modelos gráficos (ORM, UML ou CML) traduzidos posteriormente para uma linguagem orientada a objetos (PERERA et al., 2014).

3.1.2.5 Modelos Baseados em Lógica

Lógica define um conjunto de condições para que uma expressão possa ser resultante, através do processo de inferência (STRANG; LINNHOFF-POPIEN, 2004). Para descrever este conjunto de condições e expressões, há a necessidade de utilização de uma linguagem de representação formal.

Modelos baseados em lógica permitem uma representação mais rica de contexto se comparados aos modelos apresentados anteriormente, porém a inferência destes modelos é limitada e é realizada com regras bem definidas (PERERA et al., 2014).

Além disso, existem problemas relacionados a alta complexidade de motores de inferência para raciocício de regras lógicas e consequente baixa performance destes motores de inferência, além da falta de padronização de linguagens de representação de regras lógicas para interpretação das mesmas por sistemas computacionais. Estas características dificultam a reutilização e aplicabilidade deste tipo de representação em sistemas ubíquos (PERERA et al., 2014). Atualmente, são utilizadas em conjunto com outras formas de representação, como por exemplo, ontologias no formato OWL-DL.

3.1.2.6 Modelos Baseados em Ontologias

De acordo com Borst (1997), ontologia pode ser definida como uma especificação formal e explícita de um conjunto de conceitos de forma compartilhada. Informações de contexto podem ser representadas utilizando tecnologias semânticas através do uso de ontologias. Atualmente, existem diversos padrões para a representação de ontologias, tais como: RDF, RDFS, RDFa e OWL. A capacidade de representação e de inferência varia de acordo com a linguagem de representação utilizada (STRANG; LINNHOFF-POPIEN, 2004).

Existem muitas opções para a execução de inferências em ontologias. Geralmente elas estão associadas a motores de inferência, tais como: Fact++ (TSARKOV; HORROCKS, 2006), Racer (TING et al., 2010), Pellet (SIRIN et al., 2007) e Jess (FRIEDMAN-HILL, 2003). Porém, a recuperação de contexto pode ter alto custo computacional e isto interfere diretamente no desempenho dos sistemas, principalmente se for levada em consideração a alta taxa de geração e atualização de informações em sistemas ubíquos (STRANG; LINNHOFF-POPIEN, 2004).

Ontologias oferecem o suporte de padrões disseminados de representação (como OWL e RDF) e de inferência através das linguagens SWRL (*Semantic Web Rule Engine*), SQWRL

(*Semantic Web Query Language*), e SPARQL (*SPARQL Protocol and RDF Query Language*). Além disso, oferecem recursos para representações de contexto mais complexas que os modelos apresentados anteriormente, além de fornecer métodos de validação mais completos se comparada à outros modelos (PERERA et al., 2014) (BETTINI et al., 2010).

Diversas ontologias têm sido propostas para representar contextos em arquiteturas ubíquas. A ontologia CoOL (*Context Ontology Language*) (STRANG et al., 2003) foi uma das primeiras iniciativas de representação de contexto utilizando ontologias para permitir a interoperabilidade de definições entre sistemas ubíquos.

Além de ontologias de contexto genéricas[3], existem ontologias que modelam contexto em domínios específicos. A ontologia CoDAMoS (PREUVENEERS et al., 2004) representa contexto relacionado ao domínio de ambientes inteligentes. A ontologia foi definida com base nos principais conceitos relacionados ao contexto de casas inteligentes: usuário, serviço, plataforma e ambiente. A partir da definição destes conceitos na ontologia, os autores expandiram para subconceitos e definiram um conjunto de relações semânticas entre os conceitos.

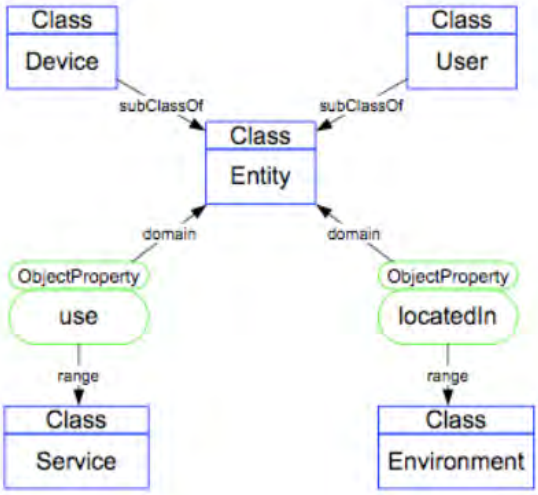
A ontologia CoBrA (CHEN et al., 2003) foi proposta para representar informações de contexto utilizadas no *framework* CoBrA. Ela é descrita na linguagem OWL-DL e define um conjunto de conceitos centrais e um conjunto de regras lógicas para a definição de casos de uso. A ontologia mIO! (POVEDA-VILLALÓN et al., 2010) foi definida como uma rede de ontologias baseada na metodologia Neon (SUÁREZ-FIGUEROA, 2010) para criação de ontologias. Esta ontologia tem como foco de modelagem o domínio de contexto para dispositivos móveis e interfaces destes dispositivos.

A ontologia PiVOn (HERVÁS et al., 2010) foi definida como uma ontologia de representação de contexto e de alta representatividade de atividades humanas se comparada a outras ontologias propostas para a representação de contexto (RODRÍGUEZ et al., 2014). Ela oferece exemplos de regras na linguagem SWRL para eliminação de ambiguidades e incremento da qualidade da informação. Além disso, foi desenvolvida sob a metodologia de redes de ontologias. Ela é composta por quatro ontologias independentes (usuários, ambiente, dispositivos e serviços). Desta forma, a ontologia pode ser estendida em partes independentes, de acordo com as necessidades do domínio. A Figura 3.6 apresenta a ligação entre as principais partes da ontologia PiVOn para a representação de informações de contexto. As classes *Service*, *Environment*, *D*evice e *User* são estendidas para outras ontologias, que juntas

[3] Ontologias de contexto são chamadas de genéricas quando representam classes genéricas de contextos e foram aplicadas em mais de um domínio de aplicação (BELLAVISTA et al., 2012).

compõem a ontologia PiVOn. Martins & Silva (2012) realizaram um estudo referente aos conceitos de ontologias propostas para a representação de contexto. A comparação foi realizada para analisar quais conceitos cada ontologia representa e o nível de expressividade de cada ontologia em relação às regras e restrições utilizadas.

Figura 3.6 – Visão geral da ligação entre as ontologias que descrevem a ontologia PiVOn.



Fonte: (HERVÁS et al., 2010).

Baseado na comparação feita por Martins & Silva (2012), foram adicionadas ontologias recentes para a representação de contextos no comparativo. O comparativo sumariza os conceitos representados em cada ontologia. De acordo com a Tabela 3.2, muitos dos conceitos das ontologias propostas possuem o mesmo significado, porém com nomes distintos, como por exemplo, as classes *Place* e *Location*, ou *User* e *Person*. Além desta característica, pode-se observar que:

(i) O padrão SWRL foi adotado para a representação de regras lógicas de primeira ordem para ontologias OWL-DL;

(ii) Ontologias recentes utilizam conceitos mais abstratos, se comparadas às primeiras ontologias de representação de contexto, tais como *Situação* e *EntidadeDeContexto;*

(iii) Em paralelo ao uso de expressões mais genéricas, há uma maior utilização de restrições e condições de construção de indivíduos em ontologias mais recentes. Isto se deve principalmente ao uso do padrão OWL-DL, com expressividade $SHOIN^{(D)}$[4];

(iv) Apenas duas das ontologias analisadas (6 e 8) representam o conceito de *Situação* e conceitos relacionados à sensibilidade à situação.

[4] Lógica descritiva similar à lógica $SHOQ^{(D)}$, com a extensão de regras inversas e restrições numéricas (HORROCKS et a., 2003).

Na Tabela 3.2 são apresentados dois tipos de informação: (i) uma lista de conceitos representados em cada ontologia (representada na parte superior da tabela) e (ii) informações gerais sobre as ontologias, como por exemplo nível de expressividade e linguagem utilizada para definição de regras de lógica de primeira ordem.

Tabela 3.2 – Principais conceitos e expressividade de ontologias de representação de contexto

Conceito/ Ontologia	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Pessoa	√	√	√		√		√		√
Lugar	√					√		√	√
Intenção	√				√				
Usuário				√		√			√
Ambiente				√	√	√		√	
Plataforma				√	√				
Serviço				√	√	√			
Localização		√			√		√		
Atividade		√			√	√		√	
Entidade		√			√				
Política			√						
Ação			√			√		√	
Agente			√						
Espaço			√			√			
Situação						√		√	
Expressividade	*ALCOIN*	*SHIF*$^{(D)}$	*SHOIN*$^{(D)}$	Taxo-nomia	*SHOIN*$^{(D)}$	*SHOIN*$^{(D)}$	*SHOIN*$^{(D)}$	*SHOIN*$^{(D)}$	*SHOIN*$^{(D)}$
Regras	-	SWRL	-	-	SWRL	SWRL	SWRL	SWRL	SWRL
Ano	2003	2004	2004	2004	2007	2010	2012	2013	2013

1: CoBrA (CHEN et al., 2003); **2:** CONON (WANG et al., 2004); **3:** SOUPA (CHEN et al., 2004); **4:** CoDAMoS (PREUVENEERS et al., 2004); **5:** SmartSpace (HUQ et al., 2007); **6:** PiVOn (HERVÁS et al., 2010); **7:** iConAwa (YILMAZ et al., 2012); **8:** (ATTARD et al., 2013); **9:** (KAYES et al., 2013).

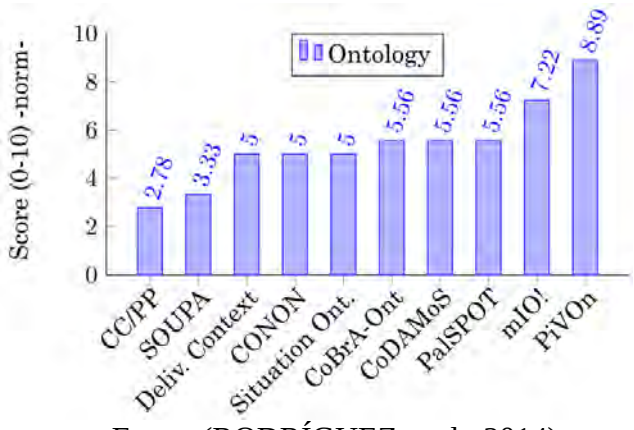
Fonte: Autoria Própria.

Rodríguez et al., (2014) realizaram um comparativo entre ontologias que representam informações de contexto e atividades. O comparativo foi realizado utilizando uma série de critérios: (*i*) Curva de aprendizado, (*ii*) Técnicas e Métodos de definição, (*iii*) Representação de interação social, (*iv*) Infraestrutura de sensores e (*v*) Escalabilidade. A Figura 3.7 apresenta o ranking de representatividade das ontologias analisadas de acordo com os critérios analisados para a representação de atividades e contextos.

As linguagens OWL e OWL 2 (versão mais recente) oferecem alta expressividade no que se refere a representação de conhecimento. Porém, elas são insuficientes para a representação de determinadas formas de contexto comuns e para a representação de relações cíclicas. Assim, as ontologias que obtiveram a maior pontuação no comparativo utilizam outras formas complementares para a representação de contextos. Estas representações

complementares geralmente são modeladas em regras de lógica de primeira ordem, utilizando as linguagens SWRL (W3C, 2004) ou SPIN (SPIN, 2016).

Figura 3.7 – Ranking de representatividade das ontologias analisadas.



Fonte: (RODRÍGUEZ et al., 2014).

3.1.3 Comparativos entre Modelagens de Contexto

Strang & Linnhoff-Popien (2004) realizaram uma comparação entre as formas de representação de contexto mais utilizadas em sistemas ubíquos. Para realizar esta comparação, foi elencado um conjunto de requisitos que sistemas ubíquos devem atender: Composição distribuída (*dc*), Validação Parcial (*pv*), Riqueza e Qualidade da Informação (*qua*), Informação Incompleta e Ambígua (*inc*), Nível de Formalidade (*for*) e Aplicação em Ambientes Existentes (*app*). A Tabela 3.3 sumariza os resultados da comparação.

Pode-se observar, na tabela resultante do comparativo, que a modelagem baseada em ontologias foi a que atendeu melhor aos requisitos listados, seguido pela modelagem orientada a objetos. A modelagem baseada em ontologias teve menor eficiência apenas na questão relacionada a aplicabilidade em sistemas ubíquos em relação aos esquemas de marcação.

Tabela 3.3 – Comparativo entre formas de representação de contexto.

Modelo / Requisito	**dc**	**pv**	**qua**	**inc**	**for**	**app**
Chave-Valor	-	-	-	-	-	+
Esquemas de Marcação	+	++	-	-	+	++
Modelos Gráficos	-	-	+	-	+	+
Modelos Orientados a Objetos	++	+	+	+	+	+
Modelos Baseados em Lógica	++	-	-	-	++	-
Modelos Baseados em Ontologias	++	++	+	+	++	+

- Não atende ao requisito

\+ Atende parcialmente o requisito

++ Atende completamente o requisito

Fonte: Tradução livre de (STRANG; LINNHOFF-POPIEN, 2004).

De acordo com Bettini et al., (2010), as modelagens de contextos para sistemas de gerenciamento de contexto em geral possuem os seguintes requisitos:

(a) **Heterogeneidade e Mobilidade**: Sistemas sensíveis ao contexto possuem fontes heterogêneas de dados. Estas fontes de dados são móveis e geram um grande volume de informações. Desta forma, é necessário que existam formas de modelar estas características e mecanismos de tratamento destas características em sistemas ubíquos;

(b) **Relacionamentos e Dependências:** Podem existir muitas relações entre elementos de contexto em sistemas ubíquos. Além disso, podem existir elementos de contexto que dependem diretamente de outros elementos. Por exemplo, a largura de banda de uma conexão de rede pode interferir diretamente na distribuição de algumas informações de contexto;

(c) **Sem restrição de tempo**: Aplicações sensíveis ao contexto podem consultar informações de contexto coletadas no passado e informações futuras (previsão) em relação a contextos. O gerenciamento destas informações é complexo, pois há uma grande quantidade de informações e as operações de atualização dos dados são constantes;

(d) **Imperfeição**: Contexto é gerado a partir de diversas fontes. Em muitos casos, a informação destes sensores é coletada ou distribuída de forma incorreta ou imprecisa. Desta forma, uma boa modelagem de contexto deve ser capaz de representar a qualidade do contexto para evitar contextos incompletos, ambíguos ou conflitantes;

(e) **Inferência**: Sistemas ubíquos sensíveis ao contexto devem tomar decisões de adaptação da sua execução ou distribuição de informação de acordo com informações de contexto. Desta forma, é fundamental que existam mecanismos de inferência que suportem a modelagem de contexto utilizada;

(f) **Usabilidade no formalismo de modelagem**: Os modelos de contexto são desenvolvidos por projetistas de sistemas sensíveis ao contexto. Portanto, uma das características desejáveis na modelagem de contexto é que existam ferramentas e metodologias que auxiliem os projetistas nestes processos;

(g) **Eficiência no acesso ao contexto**: O grande número de informações geradas, aliada a complexidade da representação de informações de contexto podem dificultar o acesso a estes dados em tempo real. Portanto, a eficiência no acesso de informações de contexto é fundamental para sistemas ubíquos.

A Tabela 3.4 apresenta uma sumarização dos resultados da comparação entre três formas de modelagem de contexto, feita por Bettini et al., (2010).

Pode-se observar na comparação que Modelos Ontológicos possuem maior capacidade de representação e inferência, porém são pouco eficientes. Esta baixa eficiência se deve principalmente à complexidade computacional envolvida na construção e execução de motores de inferência (BETTINI et al., 2010).

Tabela 3.4 – Comparativo entre formas de modelagem de contexto

	Modelo Orientado a Objetos	**Modelo Espacial**	**Modelo Ontológico**
Heterogeneidade	+	~	+
Mobilidade	~	+	-
Relacionamentos	~	~	+
Sem restrição de tempo	+	+	-
Imperfeição	~	~	-
Inferência	~	-	+
Usabilidade	+	~	~
Eficiência	~	+	-

+ Satisfatório

~ Parcialmente satisfatório

- Insatisfatório

Fonte: Tradução livre de (BETTINI et al., 2010).

Modelos espaciais são mais eficientes se comparados a ontologias, porém não possuem tanta capacidade de representação em comparação aos modelos ontológicos e baseados em objetos. Como pode-se observar na Tabela 3.4, nenhuma das formas analisadas foi satisfatória na questão de imperfeição da informação. Este problema pode ser tratado com a utilização de técnicas de inteligência artificial aplicadas em conjunto com modelos de representação para a inferência de informações imprecisas. Dentre estas técnicas destacam-se: Lógica Fuzzy, Lógica Probabilística, Redes Bayesianas ou Modelo de Hidden Markov (BETTINI et al., 2010).

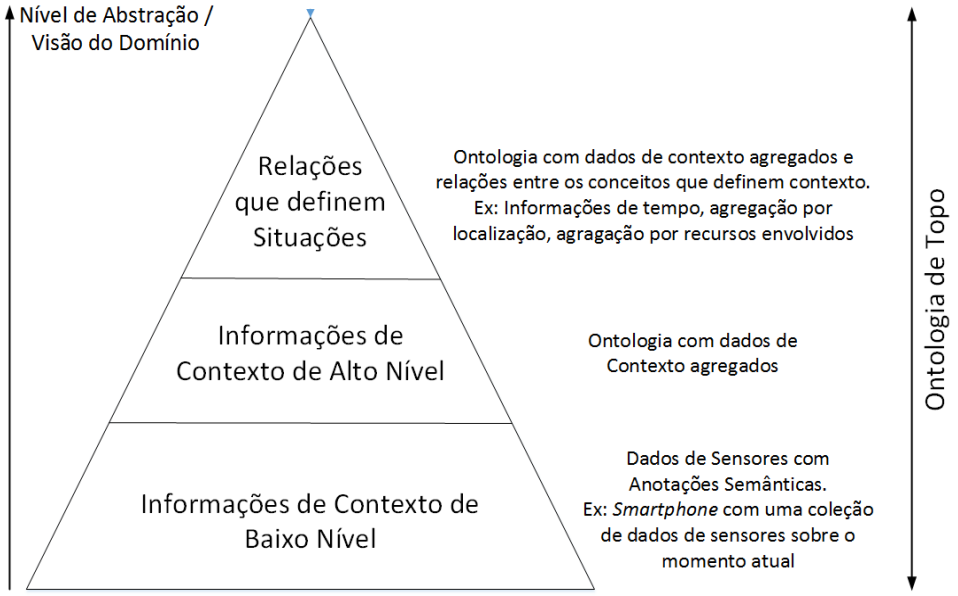
3.2 Sensibilidade à Situação

Em arquiteturas sensíveis ao contexto, situações são interpretações semânticas de um contexto de baixo nível. Esta interpretação é realizada com base em informações que agrupam conjuntos de contextos, como por exemplo a localização, tempo ou o perfil de um usuário (BETTINI et al., 2010). Isto permite com que especificações de alto nível sobre atividades humanas sejam utilizadas em compartilhadas em sistemas ubíquos.

3.2.1 Definição de Situação

De acordo com Endsley (1988) sensibilidade à situação (*Situation Awareness* – SWA) pode ser definida como *"a percepção sobre elementos do ambiente associados à localização e tempo, à compreensão sobre o significado destes elementos e à projeção do estado destes elementos em um futuro próximo"*. A Figura 3.8 apresenta os níveis de abstração de contexto utilizados em arquiteturas sensíveis ao contexto.

Figura 3.8 – Níveis de abstração de informações de contexto



Fonte: adaptado de (BETTINI et al., 2010).

No nível mais baixo (*Low-Level Context Information*) as informações de contexto são abstraídas como informações de baixo nível, próximas a representação de dados brutos que são coletados pelos sensores distribuídos no ambiente. O nível intermediário (*High-Level Context*) adiciona um nível de abstração as informações e define relacionamentos semânticos entre as informações.

A partir deste nível, as informações representadas são passíveis de reutilização por outros domínios. O terceiro nível define relacionamentos entre situações (que por sua vez definem uma abstração de contextos).

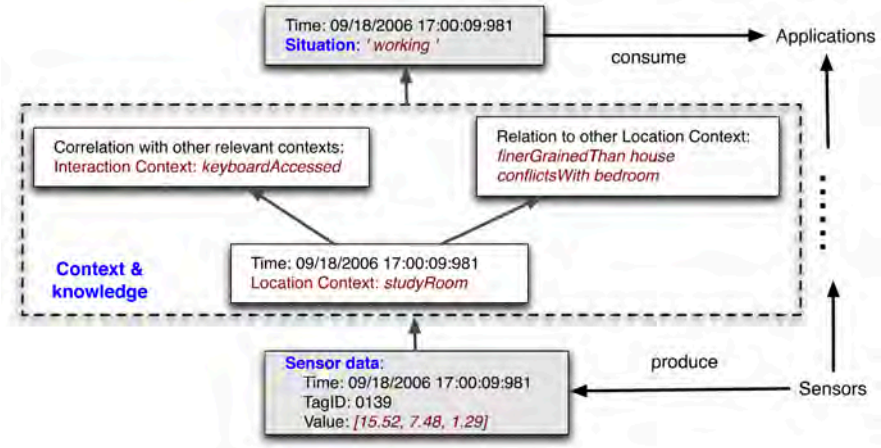
Um exemplo dos três níveis de abstração de contextos é apresentado na Figura 3.9, onde dados de sensores são coletados (primeiro nível de abstração). Estes dados são convertidos para uma abstração de alto nível de contexto, onde surgem relações semânticas

com outros conceitos. No terceiro nível, as ligações semânticas e os conceitos formam um novo conceito, definido como situação *working*, neste exemplo.

Situações possuem ligações semânticas ricas entre si, tais como (YE et al., 2012):

- **Generalização**: Uma situação pode ser especializada de outra situação. Por exemplo, podemos considerar que a situação *escrever_artigo* é uma situação especializada de trabalho. Portanto, as condições para que a situação *trabalho* seja inferida são condições para que a situação *escrever_artigo* também seja inferida;

Figura 3.9 – Exemplo de fluxo de informação em sistemas pervasivos



Fonte: (YE et al., 2012).

- **Composição**: Uma situação pode ser composta de mais situações. Por exemplo, a situação *dirigir* pode ser comporta pelas situações *movimentar_direção*, *trocar_marchas*, *freiar*, *acelerar*, entre outras;

- **Dependência**: Uma situação pode depender diretamente de outra situação se a existência da primeira estiver condicionada a existência da segunda;

- **Contradição**: A existência de uma situação em um determinado momento pode limitar (excluir) a possibilidade de existência de outra situação no mesmo intervalo de tempo. Por exemplo, as situações *tomar_banho* e *dirigir* não podem co existir ao mesmo tempo com sobre a mesma pessoa;

- **Sequência temporal**: Situações podem ocorrer durante, antes ou depois de outras situações.

Para sistemas ubíquos, há três principais campos de pesquisa que envolvem situações e sua utilização: (i) Representação de Situações; (ii) Especificação de Situações e (iii) Descoberta de Situações. Neste trabalho, aborda-se apenas aspectos relacionados a representação e especificação de situações baseados em ontologias.

3.2.2 Representação e Inferência de Situações

Em arquiteturas sensíveis ao contexto, situações são interpretações semânticas de um contexto de baixo nível. Esta interpretação é realizada com base em informações que agrupam conjuntos de contextos, como por exemplo, a localização, tempo ou o perfil de um usuário (BETTINI et al., 2010). Isto permite com que especificações de alto nível sobre atividades humanas sejam utilizadas em compartilhadas em sistemas ubíquos.

Ontologias podem ser utilizadas como formas de especificação de situações. Como informações de contexto de alto nível podem ser representadas utilizando ontologias, o conhecimento inferido sobre este contexto de alto nível pode ser representado como uma situação na própria ontologia (YE et al., 2012).

Como especificação de situações, desenvolvedores podem utilizar regras lógicas associadas a ontologias. A seguir, é apresentado um exemplo de regra que especifica a situação dormir[5].

```
(?usuario, rdf:type, socam:Pessoa),
(?usuario, socam:localizadoEm, socam:Quarto),
(?usuario, socam:temPostura, ‘DEITADO‘),
(socam:Quarto, socam:nivelDeLuz, ‘BAIXO’),
(socam:Quarto, socam:statusDaPorta, ‘FECHADO’)
-> (?usuario, socam:status, ‘DORMINDO’)
```

Diversas ontologias foram propostas como especificações de base para a definição de situações. Baumgartner & Retschitzegger (2006) realizaram um comparativo entre os principais trabalhos de ontologias para a representação de situações com base em alguns critérios importantes. São eles: (i) Conceitos de Alto Nível, (ii) Conceitos Específicos de *Situation-Awareness*, e (iii) Modelagem de características de ontologias de alto nível.

De acordo com o comparativo, a maioria das ontologias propostas atendeu aos requisitos de modelagem de conceitos. Porém, somente a ontologia SOUPA (CHEN et al., 2004) modelou questões temporais (um dos principais requisitos na modelagem de situações). Além disso, nenhuma das ontologias analisadas define de forma consistente os papéis de objetos nas situações. Os papéis definidos por (BAUMGARTNER; RETSCHITZEGGER, 2006) são:

- **Iniciador**: Controla a atividade principal de uma situação e está presente no início da situação. Exemplo: Usuário ou agente;

[5] Tradução livre da definição apresentada por Gu et al., (2005).

- **Objetivo**: Controla a atividade principal de uma situação e está presente no final da situação;

- **Recurso**: Não está ativo durante toda a situação e está presente no inicio da situação;

- **Essência**: Não está ativo durante toda a situação e está presente no final da situação.

Outra característica que não é contemplada pelas ontologias analisadas é o conceito de Tipos de Situação. Tipos de Situação definem situações genéricas sem ligação de tempo ou espaço. Desta forma são genericas o suficiente para permitir a especialização de acordo com o domínio da aplicação. Um exemplo pode ser a situação *"nevoeiro causa acidente"*.

Esta situação apenas define a relação *"causa"* entre *"nevoeiro"* e *"acidente"*. Desta forma, uma instância desta situação que contém as definições de tempo e espaço de cada um dos objetos da situação (neste caso, nevoeiro e acidente) permite com que situações sejam especificadas independentemente de domínio, onde o domínio define as questões de tempo e espaço dos objetos envolvidos (BAUMGARTNER; RETSCHITZEGGER, 2006).

3.3 Recuperação de Dados Baseada em Contexto

Informações de contexto são essenciais para sistemas ubíquos, pois o tratamento destas informações e a sua utilização permitem com que estes sistemas possam se adaptar às necessidades dos usuários e de outros sistemas. Estas adaptações devem ser feitas em tempo real e podem resultar tanto em mudança de comportamento das aplicações, a execução, quanto em na recuperação de informação, o conteúdo (BELLAVISTA et al., 2012).

Informação pode ser definida como *"uma coleção de documentos discretos, onde cada documento pode ser subdividido em um conjunto de campos"* (BROWN; JONES, 2001). A consulta de dados baseada em contexto (*Context-Aware Retrieval* – CAR) define que as operações de recuperação de informação (*Information Retrieval* – IR) e filtragem da informação (*Information Filtering* – IF) devem ser baseadas em informações de contexto informadas ao sistema no momento da consulta da informação. Desta forma, informações armazenadas, seja de forma estruturada ou não estruturada, podem ser adaptadas de acordo com o contexto informado.

Como o contexto envolve grandes conjuntos de informações, definidas em campos, ele também pode ser considerado um documento em uma coleção. Desta forma, contextos devem ser passíveis de serem consultados da mesma forma que um documento comum relacionado ao domínio (BROWN; JONES, 2001). Portanto, contexto pode ser utilizado de duas formas na recuperação de informação (BROWN; JONES, 2001):

(a) Para derivar uma consulta que retorne os documentos que melhor se enquadrem no contexto requisitado;

(b) Para tratar contexto como um documento, ou seja, o contexto se torna a fonte de informação a ser consultada.

Em relação ao item (a), a integração entre contexto e o domínio permite com que exista uma integração entre o que é conhecido (a modelagem de conhecimento relacionada ao domínio de aplicação e a aplicação em si) e o que é feito (a modelagem e uso do contexto do ambiente) por sistemas de informação (ORSI; TANCA, 2011).

Explicitamente ou implicitamente, o conhecimento modelado em sistemas computacionais possui componentes contextuais. Assim, estes componentes podem ser utilizados de alguma forma para filtrar, mudar o foco ou reduzir informações em consultas (ORSI; TANCA, 2011). As abordagens de recuperação de dados baseada em contexto podem ser divididas em grupos (COLACE et al., 2015):

(a) Utilização de diferentes perspectivas (visões) na modelagem de dados: Utiliza informações de contexto para evitar ambiguidades na interpretação de dados em condições distintas, permitindo com que entidades assumam valores distintos de acordo com o contexto. Nesta categoria estão trabalhos que propõem extensões do modelo de dados (por exemplo, o modelo relacional) e extensões de operadores de consulta para a criação de visões de acordo com informações de contexto;

(b) Particionamento de bases de dados: Utiliza informações de contexto para classificar grupos de entidades de acordo com uma informação de contexto. É uma estratégia considerada antiga e utilizada apenas pelos primeiros trabalhos na área;

(c) Escolha de serviços relevantes: Utiliza contexto dos usuários como parâmetro para comparação com informações de contexto que descrevem serviços de arquiteturas ubíquas. Desta forma, sistemas que implementam esta técnica podem executar serviços que correspondem ao contexto do usuário;

(d) Filtragem de informação: Utiliza informações de contexto para selecionar dados relevantes em bases de dados. Este tipo de abordagem está relacionada a sistemas de recomendação, geralmente com a implementação de técnicas que permitem com que usuários associem classificações para uma recomendação baseada em contexto, e caso a classificação seja considerada aceitável, recomendações semelhantes são realizadas quando o contexto for similar ao informado anteriormente.

As abordagens de recuperação de dados baseada em contexto comtemplam a execução de um conjunto de passos, separados em duas fases, que são comuns em sistemas sensíveis ao contexto (COLACE et al., 2015):

- **Fase de modelagem**: O modelo de contexto é definido de acordo com os cenários que descrevem os possíveis contextos onde a aplicação estará inserida. Após, o modelo de contexto é associado ao conhecimento relacionado a contextos de interesse. Este conhecimento está relacionado ao domínio da aplicação, como por exemplo, entidades em uma base de dados, regras de negócio, serviços relevantes, entre outros;

- **Fase de execução**: O sistema reconhece o contexto atual do ambiente e se este contexto corresponde a um dos contextos de interesse da aplicação. Se o contexto atual corresponde a um contexto previamente modelado, o conhecimento associado ao contexto é utilizado pelo sistema.

3.4 Conclusões sobre a Fundamentação Conceitual

Este capítulo apresentou a fundamentação conceitual necessária para compreensão dos próximos capítulos, relacionados ao aprofundamento na análise de trabalhos relacionados, a pesquisa apresentada nesta tese e a abordagem de consulta de dados de domínio baseada em contexto modelado em ontologias.

Como foi apresentado em Perera et al., (2014), Bettini et al., (2010) e Strang & Popien, (2004), a modelagem de contextos e situações em sistemas ubíquos apresenta uma série de desafios em relação as formas de representação e aplicação destas formas de representação. Nestes estudos, foi constatado que a representação de contextos e situações em ontologias apresenta uma série de vantagens em relação à representação de contextos em outras formas, como por exemplo, o alto nível de expressividade e a existência e manutenção de padronizações para as linguagens de representação.

Há uma série de ontologias definidas para representação de contextos em sistemas ubíquos. Estudos comparativos recentes, apresentados por Rodríguez et al., (2014) e Martins & Silva (2012) demonstraram que existem diferenças nas representações de contextos, principalmente em relação à curva de aprendizado e aplicação das ontologias em sistemas ubíquos. Neste capítulo, a comparação de Martins & Silva (2012) foi estendida no que se refere ao número de ontologias e foi possível observar que as linguagens OWL-DL e SWRL têm sido utilizadas para a representação de ontologias de contexto.

Eeste capítulo ainda apresenta a fundamentação conceitual relacionada a utilização de sensibilidade ao contexto e a situações na consulta de dados relacionados ao domínio de aplicações. No próximo capítulo é apresentada a continuação desta análise, com foco nas soluções relacionadas ao trabalho apresentado na tese.

4 ANÁLISE DE TRABALHOS RELACIONADOS

Trabalhos recentes propuseram a integração entre contexto e informações de domínio. Esta integração é realizada para permitir que informações de contexto possam ser utilizadas na filtragem de informações no momento da realização de consultas. Neste capitulo são apresentados os principais trabalhos relacionados à pesquisa realizada na tese e à abordagem de filtragem de informação, o estado da arte da área, uma análise qualitativa dos trabalhos relacionados e comparadas as principais diferenças que a pesquisa descrita nesta tese apresenta.

4.1 *Framework* HyConSC

HyConSC (ANDERSON et al., 2006) é um *framework* que realiza a integração entre informações de contexto e informações de domínio. As informações de contexto são persistidas como anotações em documentos armazenados em bancos de dados e posteriormente utilizadas na realização de consultas para recuperar documentos com contextos semelhantes ao informado no momento da consulta.

Contextos são modelados através de um *template* com um conjunto de definições e valores possíveis pré-definidos. Para realizar a consulta de contextos, os desenvolvedores informam as seguintes informações ao *framework*:

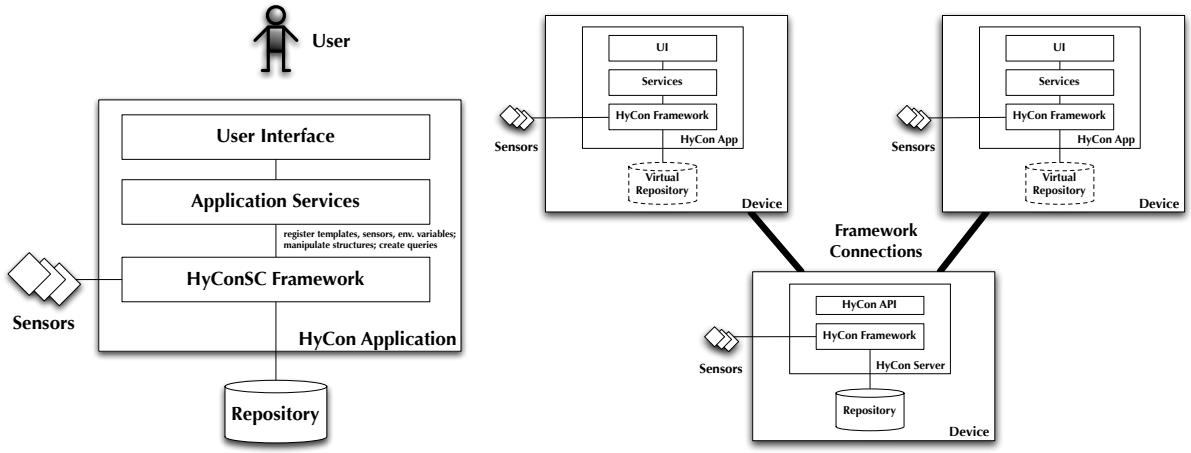
- **Nome de Propriedade**: Nome da propriedade relacionada ao contexto sobre a qual se deseja pesquisar. Exemplo: *tipo_de_propriedade*;
- **Filtro**: Uma restrição definida para o valor sobre o qual se deseja pesquisar. Exemplo: igualdade, maior, menor;
- **Valor do Alvo**: o valor da propriedade de contexto. Exemplo: *coordenada_x* = 10.

A persistência de contextos e de documentos é feita através de arquivos XML. Para realizar a recuperação dos documentos de acordo com o contexto, foi definido um algoritmo, representado na sequência de passos apresentados a seguir:

1 - Dadas as consultas A e B, onde A.numeroDeItens() ≥
2 - B.numeroDeItens()
3 - Seta *total* = 0
4 - Seta *itens* = A.numeroDeItens()
5 - Para o item A_i, em A, encontrar todos os itens em B que possuem o mesmo nome de propriedade, de B_0 a B_j
6 - Se j > 0, então calcular a similaridade entre A_i e todos os itens de B_0 a B_j
7 - Quando completar a operação, retorna total / itens

Neste algoritmo, objetos são encontrados de acordo com a comparação entre o contexto informado no momento da consulta e o contexto associado aos documentos armazenados. O *framework* prevê a possibilidade de utilização de maneira distribuída das consultas baseadas em contexto (Figura 4.1).

Figura 4.1 – Utilização do framework HyConSC em aplicações e arquiteturas distribuídas



(a) Use of the HyConSC Framework by a Single Application.

(b) HyConSC's Support for Distributed Architectures.

Fonte: (ANDERSON et al., 2006).

Para isso, cada nó de uma rede distribuída utiliza um repositório virtual de informações de contexto, que são interligados com outros nós da rede. Cada um destes repositórios virtuais armazena informações de sensores ligados ao próprio nó. Desta forma, os componentes da arquitetura se comportam de forma parecida, tanto na utilização em um nó simples (a), quanto em uma rede distribuída, com diversos nós (b).

4.2 Metodologia para Coordenação Semântica entre Contextos Conceituais e Ontologias

Uma ferramenta de recuperação de páginas web baseada em informações de contexto foi proposta por Le Grand (2009). A recuperação de informações é separada em duas etapas distintas:

(a) **Pré-tratamento de páginas web realizada *offline***: Um tratamento prévio nas páginas web é realizado para elencar um conjunto de conceitos que são associados a estas páginas. Este processo define o contexto global, ou seja, as informações de contexto associadas à páginas web. Este contexto, modelado em *Galois Lattice* (GODIN et al., 1995), é considerado estático, ou seja, não é alterado de acordo com o tempo;

(b) **Processamento online das requisições feitas pelos usuários**: Para a realização de consultas, os usuários devem informar um conjunto de conceitos pré-definidos em uma taxonomia. Desta forma, a ferramenta faz a comparação entre os conceitos previamente definidos no processo de tratamento de páginas e os conceitos informados no momento da consulta. Este processo define o contexto instantâneo, ou seja, as informações de contexto utilizadas no momento da realização das consultas.

4.3 Álgebra Relacional de Contexto

A Álgebra Relacional de Contexto (*Context Relational Algebra* - CRA) foi proposta como uma extensão da álgebra relacional tradicional, utilizada para representar consultas em bancos de dados relacionais, que oferece suporte a um modelo lógico que permite integrar informações de contexto a bancos de dados relacionais (MARTINENGHI; TORLONE, 2009).

Dimensão de contexto é definida neste trabalho como um conjunto finito de níveis de valores. Um destes níveis é definido como nível inferior ($\perp$) e outro deve ser definido como superior (T). Para todos os membros que compõem estes níveis, uma série de mapeamentos é definida. Por exemplo, um nível inferior 23/07/2009 é mapeado para outro nível e é representado como 07/2009 que, por sua vez, é mapeado para o nível superior como *Verão*.

Dado um conjunto de dimensões de contexto *D*, o esquema de contexto é definido por $C = (A_1 : l_1, ..., A_k : l_k)$, onde cada A_i é um atributo e l_i é um nível de uma dimensão de contexto. Um contexto *c* sobre o esquema de contexto *C* é o conjunto de mapeamentos que relacionam cada atributo A_i a um membro de l_i. Assim, estes esquemas de contexto e as informações de contexto em si são armazenados em tabelas.

Uma relação *C* sobre uma relação $R(X^S \parallel X^C)$ é um conjunto de tuplas X^S sobre um conjunto de atributos de contexto X^C. A Figura 4.2 apresenta um exemplo de uma relação *r1*, composta pelos atributos de domínio *Opera* e *Director*, e os atributos de contexto *Time:Day* e *Location:theater*.

Para realizar a consulta de informações baseada em contexto, foram definidas novas operações que utilizam os operadores de seleção, junção e extensão de álgebra relacional. Estas novas operações, derivadas das operações básicas definidas na álgebra relacional tradicional, podem ser ascendentes e descendentes. A Ascendência ou Descendência de uma operação é realizada sobre os níveis de contexto armazenados nas tabelas, ou seja, uma

operação de ascendência filtra o contexto de um nível baixo para alto, e uma operação de descendência realiza o processo inverso. As operações definidas no trabalho são:

Figura 4.2 – Exemplo da relação-C sob o esquema r1(*Opera: string, Director: string || Time: day, Location: theater*) e o esquema r3(*Company: string, Location: airport*)

$r_1 =$

Opera	Director	Time:day	Location:theater	
La Traviata	Abbado	11/05/2009	La Scala	$t_{1,1}$
La Bohème	Chailly	24/04/2008	Opéra	$t_{1,2}$
Turandot	Maazel	24/07/2009	Arena	$t_{1,3}$
Rigoletto	Muti	24/04/2008	La Scala	$t_{1,4}$

$r_3 =$

Company	Location:airport	
Alitalia	Villafranca	$t_{3,1}$
Air France	Roissy	$t_{3,2}$

Fonte: (LE GRAND et al., 2009).

- Extensão ascendente ($\hat{\varepsilon}$) e Extensão descendente($\check{\varepsilon}$): Uma extensão $\varepsilon^{A:l}_{A:l'}(r)$ acresenta elementos de contexto ao resultado da consulta sobre a relação *r*. São utilizados com dois atributos de contexto, *A:l* e *A:l'*, que definem os níveis de contexto que devem estar presentes na consulta. Um exemplo de extensão ascendente $\hat{\varepsilon}^{\mathsf{City}}_{\mathsf{Theater}}(r_1)$ sobre a relação *r1* é apresentado na Figura 4.3.

Figura 4.3 – Exemplo da relação-C sob o esquema r5.

$r_5 =$

Opera	Director	Time:day	Location:theater	Location:city	
La Traviata	Abbado	11/05/2009	La Scala	Milan	$t_{5,1}$
La Bohème	Chailly	24/04/2008	Opéra	Paris	$t_{5,2}$
Turandot	Maazel	24/07/2009	Arena	Verona	$t_{5,3}$
Rigoletto	Muti	24/04/2008	La Scala	Milan	$t_{5,4}$

Fonte: (LE GRAND et al., 2009).

- Seleção ascendente ($\hat{\sigma}$) e Seleção descendente ($\check{\sigma}$): As seleções definidas na CRA funcionam da mesma forma que uma seleção tradicional da álgebra relacional. A diferença é que a condição envolve um atributo de contexto em uma granularidade maior (ascendente) ou menor (descendente). Por exemplo, uma consulta $\hat{\sigma}_{\mathsf{City}=\mathtt{Milan}}(r_1)$ retornaria o conjunto {$t_{1,1}$, $t_{1,4}$}.

- Junção ascendente ($\hat{\bowtie}$) e Junção descendente ($\check{\bowtie}$): As junções definidas na CRA funcionam da mesma forma que as junções definidas na álgebra relacional. A principal diferença é que nestas junções atributos de contexto são utilizados como elemento de junção para duas relações. A Figura 4.4 apresenta um exemplo de junção $r_1 \hat{\bowtie}_{\mathsf{City}} r_3$.

Figura 4.4 – Exemplo da relação-C sob o esquema r7.

$r_7 =$

Opera	Director	Company	Time:day	Location:theater	Location:airport	
La Bohème	Chailly	Air France	24/04/2008	Opéra	Roissy	$t_{7,1}$
Turandot	Maazel	Alitalia	24/07/2009	Arena	Villafranca	$t_{7,2}$

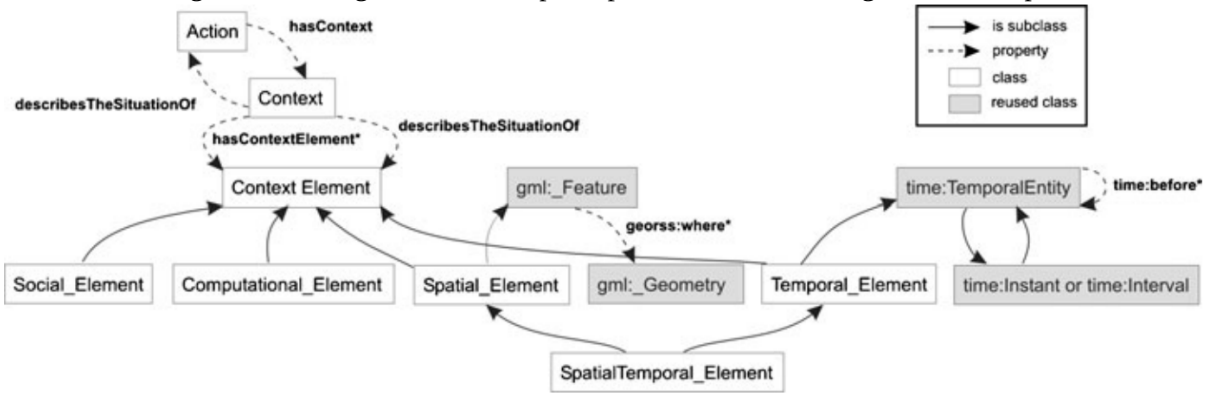
Fonte: (LE GRAND et al., 2009).

4.4 Gerenciamento Semântico e Sensível ao Contexto de Multimídia Móvel

Uma ferramenta de gerenciamento de conteúdo multimídia foi proposto por Viana et al., (2011). Esta ferramenta utiliza informações semânticas e computação móvel para definir anotações em documentos para que estes possam ser recuperados posteriormente através de informações de contexto informadas no momento da consulta.

Para realizar a anotação semântica em conteúdos multimídia, a ontologia *ContextTop* foi definida (Figura 4.5). Nesta ontologia, foram reutilizadas as definições de tempo previamente definidas na ontologia OWL-Time (GRÜNINGER, 2011). Além disso, foram definidos conceitos genéricos para representar elementos contextuais: (a) sociais, (b) computacionais, e (c) espaciais.

Figura 4.5 – Fragmento com as principais classes da ontologia *ContextTop*



Fonte: (VIANA et al., 2011).

Uma vez que documentos são inseridos, um algoritmo lê os metadados sobre o documento e os cataloga de acordo com um conjunto de regras de inferência. Além disso, os documentos são indexados de acordo com as anotações de contexto que são adicionadas a eles.

Para realizar a consulta de documentos, usuários especificam as relações semânticas entre os documentos e palavras-chave elencadas no momento da consulta. Estas palavras-chave são relacionadas a: (i) o que, (ii) onde, (iii) quem e (iv) quando. Após a enumeração de palavras-chave, um algoritmo calcula o peso de cada palavra-chave em relação às anotações semânticas dos documentos e recupera o conjunto formado pelos documentos que possuem maior similaridade com as palavras-chave informadas.

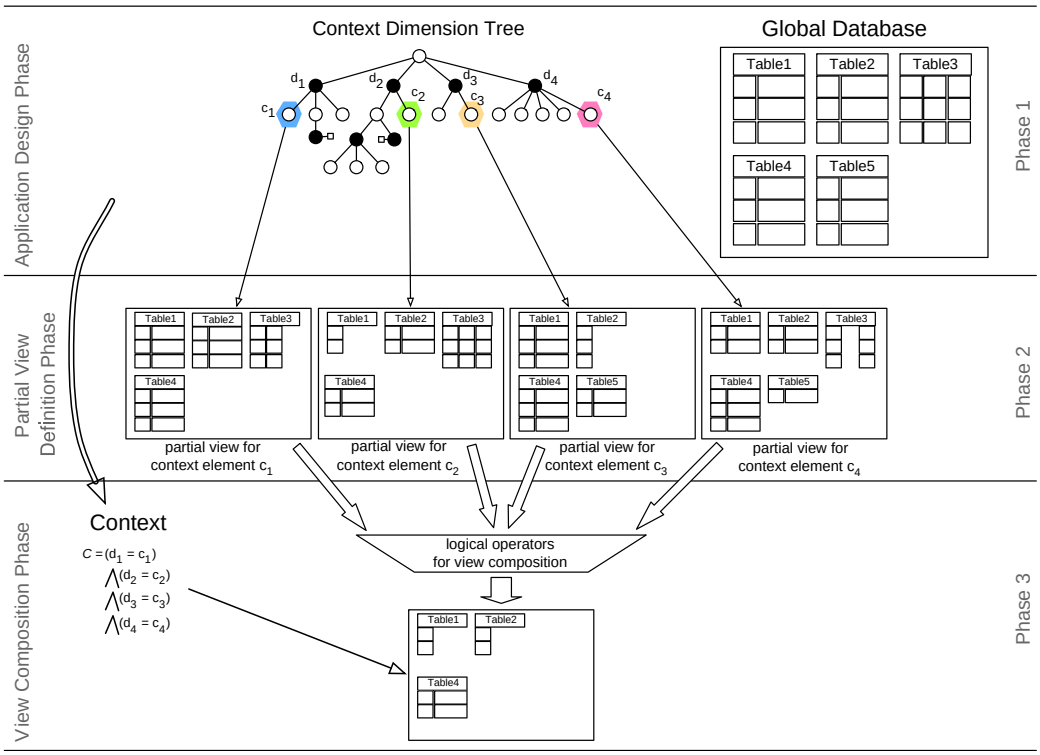
Para validar o modelo proposto, foi implementado o CoMMediA (*Context-aware Mobile and Multimedia Architecture*), uma arquitetura implementada em Java que permite

recuperar documentos multimídia informando o contexto sobre o qual se deseja saber informações (documentos associados a este contexto).

4.5 Metodologia CARVE

A metodologia CARVE (*Context-aware Automatic View Definition over Relational Databases*) foi definida como uma proposta de integração entre bancos de dados relacionais e informações de contexto na forma de um processo de geração automática de visões baseadas em contexto (BOLCHINI et al., 2013). A execução da metodologia é realizada em um conjunto de fases. A Figura 4.6 apresenta uma visão geral da metodologia e o conjunto de fases que a compõe.

Figura 4.6 – Fases da Metodologia CARVE



Fonte: (BOLCHINI et al., 2013).

A primeira fase, definida como *Fase de Design da Aplicação*, representa a fase tradicional de desenvolvimento da aplicação e do banco de dados da aplicação. Além disso, há a necessidade de definir uma *Context Dimension Tree* (CDT). Essa estrutura define os principais conceitos e indivíduos relacionados a informações de contexto que podem ser utilizados no momento da consulta de informações.

Na segunda fase, *Definição Parcial de Visões*, desenvolvedores especificam, para cada contexto, uma porção de dados do banco de dados, definida como uma visão, relacionada ao contexto. Na terceira fase, chamada de *Fase de Composição de Visões*, todas as visões são automaticamente geradas através da combinação de operadores lógicos definidos para a composição de visões. A primeira definição utilizada para a composição de visões é apresentada a seguir, onde uma expressão de adaptação e^R é definida como uma combinação de operações de projeção Π sobre um conjunto de atributos *X* e de seleção σ utilizando uma condição θ sobre a resultante da junção $\ltimes$ entre uma relação R e uma expressão de álgebra relacional *rae*.

$$e^R = \Pi_X \sigma_\theta (R \ltimes rae)$$

Estas expressões de adaptação são combinadas para formar visões baseadas em contexto. As informações de contexto são utilizadas como variáveis nas expressões. Três operadores relacionais foram definidos para a definição de visões (utilizados na segunda fase da metodologia):

- **Re-junção (RJ)**: Define uma operação de junção entre duas expressões de adaptação feitas sob a mesma origem. *R* é um esquema relacional, *X* e *Y* são conjuntos de atributos de *R*, e θ_1 e θ_2 são condições de seleção sobre os atributos de *R*. Dadas duas expressões de adaptação sobre *R*, respectivamente $e_1^R = \Pi_X \exp_1$ e $e_2^R = \Pi_Y \exp_2$, a operação de re-junção é definida por $e_1^R RJ e_2^R = \Pi_{X \cup Y}(\exp_1 \cup \exp_2)$;

- **Intersecção Dupla** ($\Cap$): Define uma operação de intersecção em pares de expressões que possuem a mesma origem. É utilizada para reduzir o número de expressões de adaptação sobre uma mesma relação. Dadas duas relações *R1*(*K1*,*A1*,*A2*) e *R2*(*K2*,*B1*,*B2*,*B3*), bem como duas expressões de adaptação, $\mathcal{A} = \{R_1, \pi_{K_2,B_1} \ \sigma_{B_1 = x} R_2\}$ e $\mathcal{B} = \{\pi_{K_2,B_2} \sigma_{B_2 = y} R_2\}$, então $\mathcal{A} \Cap \mathcal{B} = \{\pi_{K_2} \sigma_{B_1 = x \wedge B_2 = y} R_2\}$;

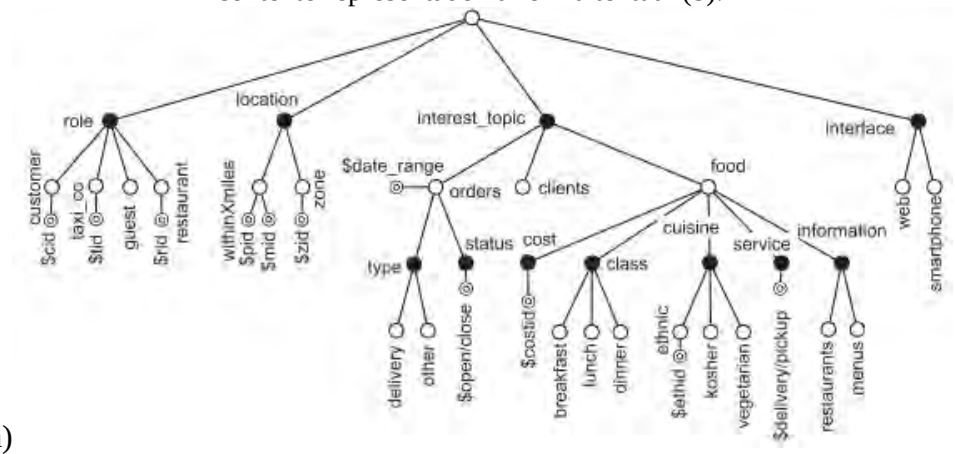
- **União Dupla** ($\Cup$): Define uma operação que retorna todas as expressões de adaptação que possuem origens diferentes das origens da outra relação.

Nesta metodologia, contexto é representado como uma árvore (Figura 4.7), onde a partir de uma raiz (◯), dimensões de contexto são definidas (●). Estas dimensões podem ser subdivididas em conceitos (◯) e podem ter um valor definido (◎).

Assim, os conceitos definidos na árvore de contexto são utilizados como operadores nas operações de seleção na definição de visões. Desta forma, visões são definidas com informações relacionadas ao contexto.

Figura 4.7 – Exemplo de *Context Dimension Tree* utilizada na metodologia CARVE (a) e de um contexto representado na forma textual (b).

a)



b)

$$C \equiv (\texttt{role} = \texttt{customer}(\$\texttt{cid} = \texttt{''Smith''})) \wedge (\texttt{interest\_topic} = \texttt{orders}) \wedge (\texttt{interface} = \texttt{smartphone})$$

Fonte: (BOLCHINI et al., 2013).

A partir da definição dos operadores relacionais e da árvore de contexto, os modeladores devem criar regras de associação entre cada elemento de contexto e uma expressão de álgebra relacional. Esta criação é feita de forma manual. A Figura 4.8 apresenta um exemplo de associação entre o elemento de contexto *interest_topic* e o conjunto de tabelas relacionados a este elemento. Assim, quando em uma consulta este elemento de contexto for informado (os elementos de contexto precisam ser informados explicitamente pela aplicação), apenas o conjunto de relações especificados a este elemento de contexto são utilizados.

Figura 4.8 – Exemplo de associação entre elemento de contexto e relações na metodologia CARVE.

$$\mathcal{R}el((\texttt{interest\_topic} = \texttt{food})) = \{\textsc{Cuisines}, \textsc{DailySpecials}, \textsc{Dishes}, \textsc{DishCategory}, \textsc{DishIngredient}, \textsc{Drinks}, \textsc{Ingredients}, \textsc{Menus}, \textsc{MenuDishes}, \textsc{Restaurants}, \textsc{RestaurantCuisine}, \textsc{RestaurantDrink}. \textsc{Pickup}. \textsc{Zones}\}$$

Fonte: (BOLCHINI et al., 2013).

A partir da definição das associações entre contextos e relações, os operadores relacionais previamente apresentados e definidos pela metodologia são utilizados para unir as visões parciais.

4.6 Sistema de Recomendação Sensível ao Contexto Influenciado por Dados de Saúde de Pacientes

A HARE (*Time-Aware, Location-Aware and Health-Aware Recommender*) é uma aplicação de recomendação de conteúdo (comerciais, páginas web, receitas, entre outros) que utiliza dados de tempo, localização e dados de saúde do paciente para a realização destas recomendações (LÓPEZ-NORES et al., 2013). Para realizar a recomendação de conteúdos, ontologias foram utilizadas para descrever os metadados sobre estes conteúdos. Em relação aos usuários, as seguintes informações são utilizadas no processo de recomendação:

(a) Informações demográficas, como idade, gênero e estado civil;

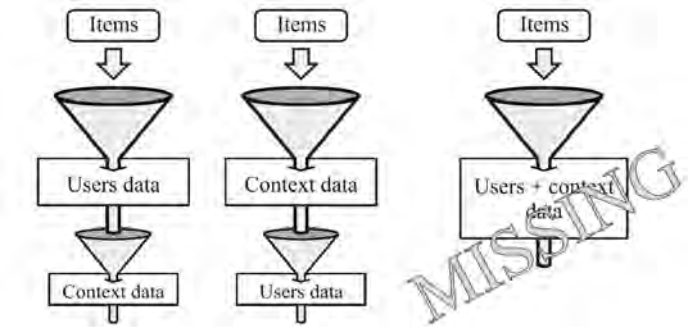
(b) Histórico de navegação em sites e compras;

(c) Histórico de condições de saúde e doenças registradas no prontuário médico do usuário.

Para acessar as informações de saúde do paciente, a aplicação se comunica com uma arquitetura que lê as informações de prontuários descritos no padrão OpenEHR (KALRA et al., 2005). Estas informações de saúde são integradas às representações semânticas que descrevem os conteúdos, a ontologia OWL-Time, utilizada para a representação de questões relacionadas à tempo e a informações de localização descritas em outra ontologia.

Geralmente, sistemas de recomendação realizam o processamento da informação relacionada à contexto primeiro, e depois, realizam o processamento sobre informações do usuário para filtrar os itens no processo de recomendação (Figura 4.9). Na aplicação HARE, uma nova forma de recomendação foi implementada de modo que as informações de usuários, seus históricos e informações de contexto fossem processadas paralelamente ao processo de recomendação.

Figura 4.9 – Métodos de gerenciamento de informações sobre usuários e contextos em sistemas de recomendação.

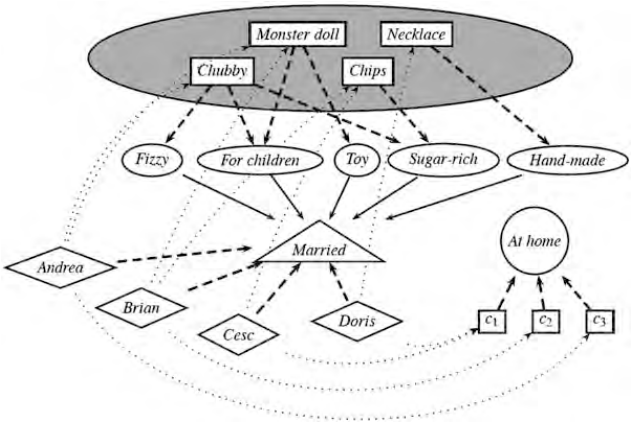


Fonte: (LÓPEZ-NORES et al., 2013).

A estratégia *Property-Based Collaborative Filtering* (PBCF) foi proposta para permitir com que informações de contextos e de usuários possam ser processadas ao mesmo tempo no momento da recomendação. Para isso, ela sempre mantém links entre as propriedades semânticas que caracterizam itens, usuários e contextos. Desta forma, consegue-se representar conhecimento que pode ser representado em regras envolvendo estas três dimensões. Um exemplo desta forma de filtragem é apresentado na Figura 4.10.

Cada ligação entre contextos, usuários e itens tem um peso. Deste modo, a combinação de pesos entre as ligações determina quais itens devem ser recomendados de acordo com um usuário ou contexto específico.

Figura 4.10 – Representação Conceitual do PBCF. Losangos: Usuários, Triângulos: Propriedades de usuários, Retângulos: Itens ou Contextos, Elipses: Propriedades de itens, e Círculos: Propriedades de Contexto



Fonte: (LÓPEZ-NORES et al., 2013).

4.7 Definição Dinâmica de Visões em Bancos de Dados Relacionais Baseada em Ontologias

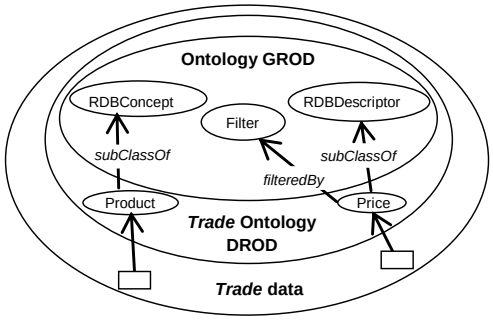
Um modelo foi proposto por Ponce-Toste et al., (2013) para realizar a representação semântica de bancos de dados relacionais e fornecer suporte à recuperação de informação com informações semânticas sobre os dados armazenados. Este modelo possui uma estrutura (Figura 4.11) baseada em um conjunto de ontologias, separadas em níveis e definidas como:

(a) **Ontologia GROD** (*General Relational Database Ontology*): Define conceitos gerais relevantes para a realização de consultas em bancos de dados relacionais;

(b) **Ontologia DROD** (*Domain Specific Relational Database Ontology*): Estende os conceitos da ontologia GROD para representar conceitos específicos do domínio da aplicação na qual o banco de dados está relacionado;

(c) **Instâncias DROD**: São instâncias que se referem diretamente aos dados persistidos no banco de dados. Neste nível do modelo, consultas em SQL mapeiam a camada sintática (os dados persistidos no banco de dados relacional) e a camada semântica (os conceitos descritos nas ontologias DROD e GROD).

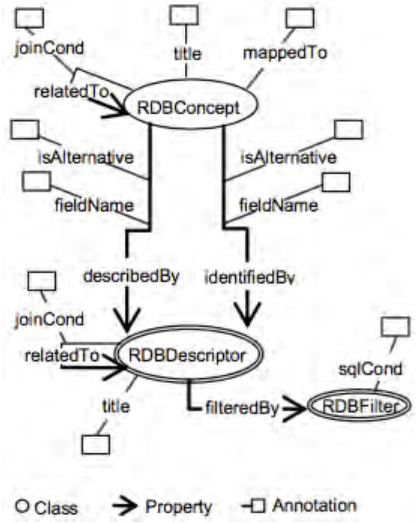
Figura 4.11 – Estrutura do modelo composto por ontologias



Fonte: (PONCE-TOSTE et al., 2013).

Para realizar a consulta de informações, foi definida uma arquitetura que implementa o modelo proposto. Na demonstração do protótipo uma interface apresenta os conceitos que representam os dados armazenados no banco de dados e opções de filtragem de informações, como por exemplo, igualdade e comparações de maior e menor valor. Assim, as consultas são traduzidas para a linguagem SQL. A Figura 4.12 apresenta a ontologia GROD.

Figura 4.12 – Estrutura da ontololgia GROD



Fonte: (PONCE-TOSTE et al., 2013).

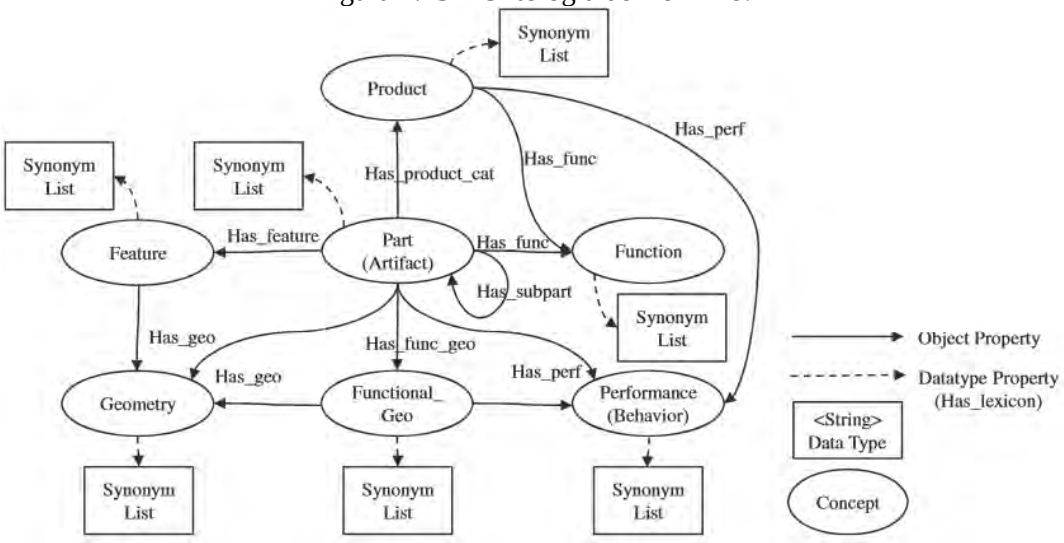
A partir da definição desta ontologia, cada indivíduo da classe RDBConcept é traduzido para uma expressão na linguagem SQL, e quando um filtro é aplicado na busca por indivíduos, a expressão em SQL correspondente é utilizada para a realização da consulta no banco de dados.

4.8 Expansão de Consultas para Personalização de Documentos de Engenharia

Uma arquitetura foi proposta por Hahm et al (2014) para realizar a recuperação personalizada de documentos de engenharia baseada na análise de perfil de usuários. A arquitetura é baseada em quatro componentes: (i) Construção de uma ontologia de domínio, (ii) indexação semântica, (iii) aprendizagem sobre o perfil do usuário, e (iv) recuperação personalizada de informação.

A definição da ontologia de domínio foi realizada para representar os principais documentos de engenharia utilizados e as principais informações sobre estes documentos. As principais classes da ontologia são apresentadas na Figura 4.13.

Figura 4.13 – Ontologia de Domínio.



Fonte: (HAHM et al., 2014).

Após a definição da ontologia de domínio, foi desenvolvido um algoritmo que extrai informações de documentos e indexa estas informações na ontologia. A indexação é feita de forma automática, o algoritmo extrai as palavras chave e verifica se elas correspondem a algum termo definido na ontologia de domínio. Com as informações indexadas, é possível realizar uma expansão de termos utilizados na consulta, ou seja, quando uma consulta é

realizada, um algoritmo procura relações entre os termos pesquisados pelo usuário e termos definidos no perfil do usuário.

Assim, os termos relacionados são incluídos na consulta e mostraram uma melhora significativa na precisão do processo de recuperação de informação. Porém a consulta é realizada somente através da marcação de termos de interesse da ontologia, e não através de uma linguagem de consulta padronizada como, por exemplo, as linguagens SQL ou SPARQL.

4.9 Análise dos Trabalhos Relacionados

Para realizar a análise sobre o estado da arte na área de recuperação de dados baseada em contexto, foi enumerada uma lista de características que foram utilizadas como critérios de análise dos trabalhos. Em relação à modelagem de contexto e situação:

- **Modelo de Contexto**: A forma com que informações de contexto são modeladas. Elas podem ser modeladas de forma relacional (***BDR***), orientada a objetos (***OO***), Baseadas em Lógica (***BL***), modelos gráficos (***G***) ou modeladas em ontologias (***O***);
- **Modelagem de Contexto para Sistemas Ubíquos**: Se a modelagem de contexto é feita com a utilização de ontologias, deve-se verificar se a modelagem é baseada em uma ontologia de contexto genérico, ou seja, independente de aplicação. Ainda, foi verificado se a modelagem de contexto foi utilizada previamente em algum trabalho relacionado a arquiteturas ou middlewares ubíquos. Para verificar se a ontologia utilizada no trabalho foi utilizada em alguma arquitetura ubíqua ou trabalho relacionado a middlewares ubíquos, foi realizada uma pesquisa nas principais bases de dados e indexadores de artigos (*ACM DL*[6], *IEEE Xplore*[7], *Science Direct*[8] e *Google Scholar*[9]) utilizando o nome da ontologia e os termos *ubiquitous*, *middleware* e *architecture*;
- **Conceitos de Contexto**: Contexto pode ser modelado e utilizado em vários níveis em sistemas ubíquos. Seguindo as definições de Dey et al., (2001) e Zimmermann et al., (2007), as principais dimensões de contexto são: Usuário (***US***), Dispositivo (***DISP***), Ambiente (***ENV***) e Serviços (***SRV***). Além disso, foi adicionado ao comparativo a

[6] http://dl.acm.org/

[7] http://ieeexplore.ieee.org/Xplore/home.jsp

[8] http://www.sciencedirect.com/

[9] https://scholar.google.com.br/

modelagem de atividades (***ATV***), que está relacionada a modelagem e inferência de situações;

- **Modelagem de Situação**[10]: Foi analisado se o trabalho leva em consideração algum mecanismo, seja ele pré-definido ou implementado no trabalho, para a definição de situações.

Em relação a modelagem de domínio da aplicação, foram analisadas as seguintes características:

- **Modelo de Informação de Domínio**: A forma com que as informações de domínio são modeladas e utilizadas na integração com informações de contexto. Elas podem ser modeladas de forma relacional (***BDR***), como documentos com anotações semânticas (***DAS***) ou como ontologias de domínio (***O***);
- **Domínio específico**: Foi analisado se o trabalho foi implementado para uma aplicação específica, ou para um domínio de conhecimento específico;
- **Utilização de Bancos de Dados**: A persistência e a recuperação de informações podem ser feitas com o auxílio de SGBDs. Alguns trabalhos não especificam (***NE***) o uso de SGBDs, outros utilizam modelos como *Triple Stores* (***TS***) ou bancos de dados relacionais (***BDR***). Alguns trabalhos não utilizam bancos de dados (***NU***).

Em relação à integração entre informações de contexto e informações de domínio, foram analisadas as seguintes características:

- **Modo de Integração**: O modo como a integração entre informações de contexto e dados de domínio é realizada. Podem ser baseadas em alinhamento de ontologias (***A***), extensões de álgebra relacional (***AR***), algoritmos de integração por análise de propriedades (***ALG***);
- **Linguagem de Consulta**: Após a realização da integração entre informações de domínio e informações de contexto, há a necessidade de definição de uma linguagem de consulta de informações, para que as informações de domínio possam ser recuperadas de forma contextualizada. A recuperação dos dados pode ser definida através de extensões de álgebra relacional (***AR***), de consultas definidas na linguagem SQL (***S***), SQWRL (***SQ***) ou SPARQL (***SP***).
- **Automatização do processo**: Foi analisado se o trabalho implementa alguma forma de automatização do processo de integração através de inferência, regras de lógica de primeira ordem ou algoritmos.

[10] Situações são definidas como interpretações semânticas de um contexto de baixo nível (YE et al., 2011).

Baseado na lista definida acima foi realizada uma análise dos trabalhos que representam o estado-da-arte na área. A taxonomia utilizada na Tabela 4.2 é apresentada a seguir, na Tabela 4.1.

Tabela 4.1 – Taxonomia utilizada no comparativo

Dimensão	Taxonomia	Abreviação	Descrição
Informações Gerais	Tipo	WSP	Workshop
		JN	Journal
		CAP	Capítulo de Livro
		CNF	Conferência
	Ano	-	-
Contexto	Modelo	BDR	Modelo Relacional
		OO	Orientado a Objeto
		BL	Baseado em Lógica
		G	Gráfico
		O	Ontologia
	Utilizado em Arquitetura Ubíqua	√	Utilizado em arquiteturas ubíquas
		-	Não utilizado em arquiteturas ubíquas
	Conceitos Representados	US	Contexto de usuário
		ATV	Representação de Atividades
		DISP	Contexto de dispositivos
		ENV	Informações de ambiente e localização
		SRV	Informações sobre serviços
	Modelagem de Situação	√	Modela conceitos relacionados a situação
		-	Não modela conceitos relacionados a situação
Domínio	Modelo	BDR	Banco de dados Relacional
		DAS	Documentos com anotações semânticas
		O	Ontologia
		OO	Orientado a Objeto
	Específico para um domínio ou aplicação	√	Específico para um domínio ou aplicação
		-	Não específico para um domínio ou aplicação
	Banco de dados	NE	Não especificado
		TS	*Triple Stores*
		BDR	Banco de dados relacional
		NU	Não utiliza
Integração	Modo de Integração	A	Alinhamento de Ontologias
		AR	Extensões de Álgebra Relacional
		ALG	Algoritmos de integração por análise de propriedades
	Linguagem de Consulta	AR	Extensões de álgebra relacional
		S	SQL
		SQ	SQWRL
		SP	SPARQL
		NP	Forma de consulta não padronizada
	Modo de Consulta	TRAD	Utiliza a linguagem de consulta de forma padronizada, sem a necessidade de alteração na consulta
		EXT	Altera a forma com que a consulta é originalmente realizada
		NU	Não utiliza linguagem de consulta padronizada
	Automatização do Processo	√	Possui algum mecanismo de automatização em alguma parte da integração
		-	Integração feita inteiramente de forma manual

Fonte: Autoria Própria.

O resultado da análise qualitativa dos trabalhos relacionados à pesquisa desta tese é apresentado na Tabela 4.2.

Tabela 4.2 – Análise qualitativa dos trabalhos relacionados

Característica / Trabalho		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Informações Gerais	Tipo	CNF	CAP	JN	JN	JN	CAP	WSP	JN
	Ano	2006	2009	2009	2011	2013	2013	2013	2014
Contexto	Modelo	G	O	BDR	O	G	O	O	O
	Utilizado em Arquitetura Ubíqua	-	-	-	-	-	-	-	-
	Conceitos Representados	DISP+ ENV	US	ENV	US+ ENV+ DISP	US+ ENV+ DISP	US+ ENV+ DISP	-	US+ ENV
	Modelagem de Situação	-	-	-	-	-	-	-	-
Domínio	Modelo	OO	O	BDR	O + DAS	BDR	O	BDR	O + DAS
	Específico para um domínio ou aplicação	-	-	-	√	-	√	-	√
	Banco de dados	NE	NU	BDR	NE	BDR	TS	BDR + OWL	NU
Integração	Modo de Integração	ALG	ALG	AR	A	AR	A	A	A
	Linguagem de Consulta	NP	NP	AR	NP	AR	SP	S	NP
	Modo de Consulta	NU	NU	EXT	NU	EXT	EXT	EXT	NU
	Automatização do Processo	√	-	-	√	√	√	-	√

1: (ANDERSON et al., 2006); **2:** (LE GRAND., 2009); **3:** (MARTINENGHI; TORLONE, 2009); **4:** (VIANA et al., 2011); **5:** (BOLCHINI et al., 2013); **6:** (LÓPEZ-NORES et al., 2013); **7:** (PONCE-TOSTE et al., 2013); **8:** (HAHM et al., 2014)

Fonte: Autoria Própria.

Na análise qualitativa sobre os trabalhos relacionados, pode-se agrupar os trabalhos em duas abordagens principais e distintas: (*i*) Trabalhos que utilizam a modelagem de contexto e de domínio em ontologias ou documentos com anotações semânticas: a modelagem de contexto e do domínio da aplicação são feitos em ontologias, que são posteriormente alinhadas. Isto permite que definições feitas na ontologia que representa o domínio possam ser recuperadas utilizando informações definidas na ontologia de contexto; (*ii*) Trabalhos que utilizam a modelagem de contexto e de domínio em modelos gráficos ou relacionais: a modelagem de contexto e do domínio é feita em modelos relacionais, em modelos gráficos, ou modelos híbridos (relacionais e gráficos). Para realizar as consultas em informações de domínio baseando-se em informações de contexto, estes trabalhos utilizam extensões de álgebra relacional.

Em relação a modelagem de contexto e de situações, pode-se observar que nenhum dos trabalhos pesquisados utiliza a modelagem de contexto baseada em ontologias genéricas de contexto, utilizadas em sistemas ubíquos (alguma das ontologias apresentadas na seção 3.2.1.6). Isto acarreta no problema de compartilhamento de contexto entre sistemas ubíquos, evidenciado em pesquisas sobre sistemas ubíquos (PERERA et al., 2014) (MAKRIS et al., 2013). Além disso, as ontologias utilizadas nestes trabalhos não representam todas as dimensões de contexto utilizadas em sistemas ubíquos, e não suportam mecanismos de inferência à situação.

Em relação aos modelos de integração para a realização de consultas contextualizadas, observa-se que nenhum dos trabalhos implementa o modelo de integração sem a necessidade de reescrita das consultas realizadas nos bancos de dados, quando são utilizados bancos de dados relacionais.

4.10 Relacionamento Entre o Trabalho Proposto e os Demais

Como apresentado anteriormente no cenário motivador (Capitulo 2) e em pesquisas (ORSI; TANCA, 2011) (TANCA et al., 2011) (BOLCHINI et al., 2013), a integração entre contexto e consultas em bancos de dados relacionais é algo relevante em sistemas atuais, principalmente no que se refere a introdução de tecnologias ubíquas nestes sistemas. O cenário motivador apresentado no capítulo 2 ainda esclarece a necessidade de características importantes que devem ser contempladas pelos modelos de integração:

(a) A modelagem de contexto deve oferecer a expressividade necessária para representar o contexto de ambientes ubíquous. Ontologias são capazes de oferecer este nível de expressividade (evidenciado pelo estudo apresentado na Seção 3.1.3). Diversas ontologias foram modeladas para a representação de contextos e situações, mas é necessário que estas ontologias tenham sido testadas e validadas em ambientes ubíquos (evidenciado pelo estudo apresentado na Seção 3.1.2.6).

(b) O foco desta tese é a integração de contexto com modelos relacionais. Isto deve-se a necessidade da realização desta integração em bancos de dados relacionais previamente modelados e que são utilizados por sistemas considerados não ubíquos (exemplos foram apresentados no capitulo 2). Desta forma, a integração entre contexto e informações de domínio deve permitir que estas informações sejam recuperadas utilizando uma linguagem padronizada de consulta pelos sistemas que acessam estas informações. Além disso, as

consultas utilizadas por estes sistemas e o tratamento destas consultas já foram previamente definidos. Assim, o *framework* de integração deve permitir a integração de contexto, mas sem a necessidade de reescrita das consultas previamente definidas (evidenciado na descrição do cenário motivador no capítulo 2).

(c) O gerenciamento de contexto envolve uma série de desafios (BELLAVISTA et al., 2012). Um dos principais é que o contexto varia constantemente e contextos de interesse[11] podem ser formados por informações distintas de acordo com a disponibilidade das fontes de informação. Desta forma, há a necessidade de que existam formas de automatizar totalmente ou parcialmente o processo de integração, pois a quantidade de informações de contexto utilizadas na personalização de consultas é considerada grande demais para ser gerenciada de forma totalmente manual (BOLCHINI et al., 2013).

A Tabela 4.3 apresenta uma relação entre os trabalhos relacionados e as características necessárias para a integração efetiva entre informações de domínio, modeladas em bancos de dados relacionais, e informações de contexto, utilizadas por sistemas ubíquos.

Tabela 4.3 – Comparativo entre trabalhos relacionados e o trabalho apresentado nesta tese

Característica / Trabalho		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
(a) Contexto	Nível de expressividade	-	√	-	√	-	√	-	√
	Utilizado em arquiteturas ubíquas	-	-	-	-	-	-	-	-
	Modelagem de Situação	-	-	-	-	-	-	-	-
(b) Domínio	Modelo Relacional	-	-	√	-	√	-	√	-
	Linguagem de consulta padronizada para bancos relacionais	-	-	√	-	√	-	√	-
(c) Integração	Mantém consultas e esquema do banco originalmente como foram pré definidos	-	-	-	√	-	-	-	√
	Automatização do Processo	Parcial	-	-	√	√	√	-	√

1: (ANDERSON et al., 2006); **2:** (LE GRAND., 2009); **3:** (MARTINENGHI; TORLONE, 2009); **4:** (VIANA et al., 2011); **5:** (BOLCHINI et al., 2013); **6:** (LÓPEZ-NORES et al., 2013); **7:** (PONCE-TOSTE et al., 2013); **8:** (HAHM et al., 2014)

Fonte: Autoria Própria.

Pode-se observar que alguns dos trabalhos relacionados utilizam modelagem de contexto com expressividade necessária para ser utilizada por sistemas ubíquos. Porém nenhum destes trabalhos utiliza modelagem de ambientes ubíquos. Isto é evidenciado pelo

[11] *"É um subconjunto ou fatia (não precisa ser sensível ao tempo) de uma parte do contexto, onde aplicações teriam interesse em uma parte específica do ambiente"* (MACHADO, 2015).

fato de que nenhuma das ontologias utilizadas na integração pelos trabalhos relacionados e utilizada por alguma arquitetura ou sistema ubíquo. Além disso, nenhuma das representações de contexto utilizadas levou em consideração características relativas à modelagem de situação.

Entre os trabalhos que utilizam expressividade considerada suficiente para a modelagem de contexto, não há abordagens que utilizem a modelagem de domínio em bancos de dados relacionais, em conjunto com a utilização de linguagens padronizadas para a recuperação de dados modelados de forma relacional.

Em relação à necessidade de reescrita das consultas, observa-se que todas as abordagens que utilizam a modelagem de domínio de forma relacional descrevem extensões das linguagens de consulta, o que acarreta na necessidade de reescrita de consultas previamente definidas para a integração de contexto. Isto se torna um problema para sistemas e bancos de dados legados (evidenciado no cenário apresentado no capítulo 2), pois aliado à necessidade de reescrita das consultas, está a necessidade de modificação nos sistemas para o tratamento das consultas modificadas.

5 FRAMEWORK PARA CONSULTA DE DADOS RELACIONAIS DE DOMÍNIO BASEADA EM CONTEXTO MODELADO EM ONTOLOGIAS

Neste capítulo é apresentado o *framework* desenvolvido nesta tese. Inicialmente, é apresentada a estrutura geral do *framework*, com a descrição dos componentes (seção 5.1). Após, são apresentadas as modelagens dos componentes que fazem parte do *framework* (seção 5.2). Estes componentes são classificados em componentes relacionados a contexto, componentes relacionados a domínio e componentes associados ao processo de alinhamento de conceitos. Na seção 5.3 é apresentada a forma de persistência e recuperação das informações utilizadas pelo *framework*. Na seção 5.4 é apresentado o processo de extensão de consultas utilizado no *framework* para contextualização de informações. Posteriormente, é apresentada a metodologia desenvolvida para a implementação do *framework* (seção 5.5) – a metodologia é utilizada para a implementação do *framework* no cenário de avaliação no capítulo 6. Já na seção 5.6 são apresentadas as considerações finais do capítulo.

5.1 Estrutura do *Framework*

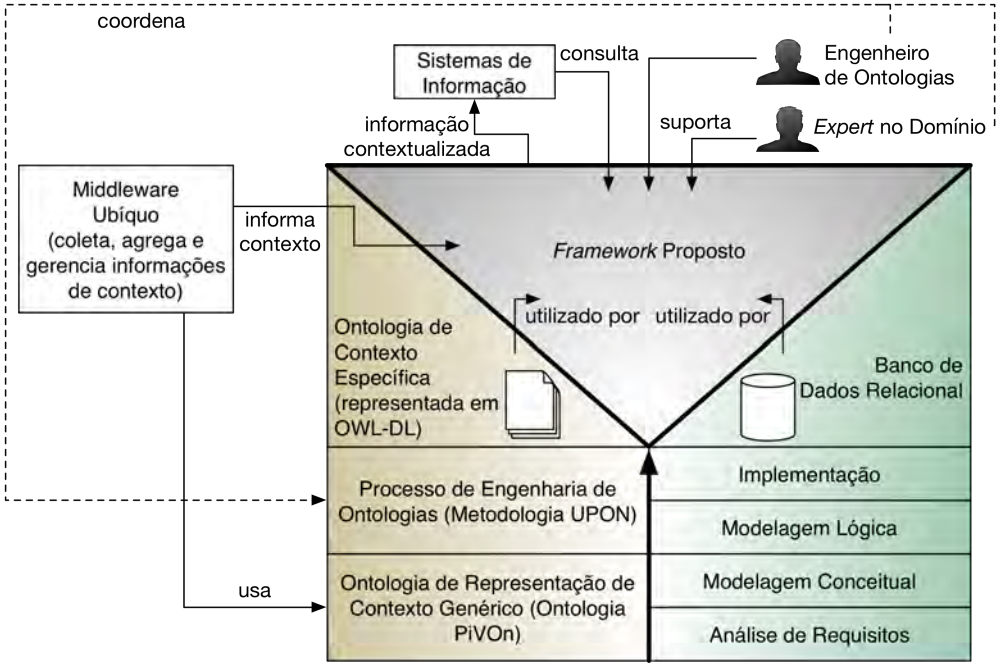
Sistemas de informação frequentemente utilizam bancos de dados relacionais como fonte de informação. O processo de modelagem destes bancos de dados é realizado para que a estrutura do banco contemple os requisitos da aplicação e do domínio no qual a aplicação está inserida (BOLCHINI et al., 2013). Em relação à modelagem de contexto, arquiteturas ubíquas frequentemente utilizam modelagens de contexto baseadas em ontologias (MAKRIS et al., 2013) (PERERA et al., 2014).

Desta forma, faz-se necessária a utilização de uma série de recursos para a utilização do *framework*. Os recursos necessários e as formas de utilização destes recursos pelo *framework* são apresentados na Figura 5.1. Pressupõe-se que um banco de dados relacional normalizado até a 3ª forma normal (3FN) tenha sido definido utilizando a metodologia tradicional de modelagem de banco de dados, composta pelas etapas de análise de requisitos, modelagem conceitual, criação do modelo lógico e implementação física (DATE; DARWEN, 1997). Para recuperar dados deste banco de dados, sistemas de informação utilizam consultas definidas na linguagem SQL.

Em relação à modelagem de contexto, sistemas considerados ubíquos frequentemente utilizam ontologias para a representação de contextos. Para a utilização da modelagem de

contexto de outro sistema, o *framework* prevê a adoção da ontologia PiVOn (HERVÁS et al., 2011) como ontologia de contexto genérica. A escolha da ontologia foi realizada pela série de vantagens em relação a outras ontologias relacionadas (um estudo mais aprofundado foi previamente apresentado na seção 3.1.3 e na comparação apresentada por Rodriguez et al., (2014)). Esta ontologia é estendida através de um processo de engenharia de ontologias. Para a extensão de ontologias, recomenda-se a utilização da metodologia UPON (NICOLA et al., 2009). Esta metodologia é utilizada por várias arquiteturas ubíquas para a extensão de ontologias, e alguns de seus processos foram utilizados na metodologia proposta para a implementação do *framework*. Após a extensão da ontologia de contexto para o domínio de aplicação do sistema ubíquo, a ontologia resultante pode ser integrada e utilizada pelo *framework.*

Figura 5.1 – Recursos necessários para implementação do *framework*.



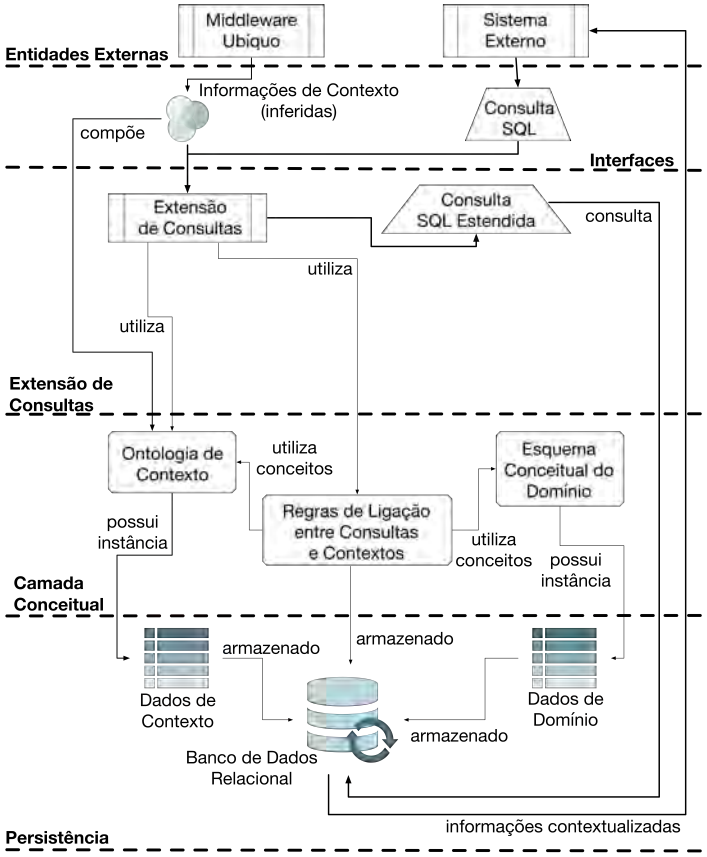
Fonte: Autoria Própria.

Para a execução dos processos previstos no *framework* para a realização de consultas, há a necessidade de suporte de *experts* no domínio de aplicação e de engenheiros de ontologias. Após a definição de regras de ligação, determinadas previamente, sistemas que realizam consultas no banco de dados relacional utilizado pelo *framework* recebem resultados de consultas de forma contextualizada. Assim, a implementação do *framework* permite que informações de contexto, modeladas em ontologias com alta expressividade possam ser utilizadas por sistemas que utilizam bancos de dados relacionais legados, sem a necessidade

de modificação das consultas originalmente definidas. A visão geral do *framework* é apresentada na Figura 5.2. O *framework* é dividido em 5 níveis:

(a) Entidades Externas: *Middlewares* ubíquos realizam o gerenciamento de informações de contexto e de recursos do ambiente. Ainda nestes ambientes, sistemas externos podem utilizar recursos provenientes de *middlewares*. Como middlewares ubíquos gerenciam informações de contexto, estes podem ser requisitados a informar ao *framework* o contexto atual do ambiente no momento da realização de uma consulta. Sistemas externos realizam consultas aos dados utilizando o *framework* que retorna um conjunto de dados que estejam de acordo com os filtros informados pelos sistemas externos e ao contexto informado pelo *middleware* ubíquo;

Figura 5.2 – *Framework* de consulta de informações de domínio baseada em contexto.



Fonte: Autoria Própria.

(b) Interfaces: Para se comunicar com o *framework*, foi definido que *middlewares* ubíquos devem utilizar interfaces REST (GRALLA, 2003), informando o contexto através de representações na linguagem XML. Sistemas externos, por sua vez, realizam a consulta de informações utilizando a linguagem SQL. As interfaces são descritas com detalhes na seção 5.4;

(c) Extensão de Consultas: A consulta realizada por sistemas através do formato SQL é estendida através de um algoritmo[12]. Este processo realiza a consulta na base de regras de ligação entre contexto e domínio e verifica se existe relações que podem ser utilizadas para a extensão da consulta em SQL. Neste processo, são utilizadas as definições das ontologias que descrevem contexto, que descrevem o esquema de banco de dados do domínio e as definições de alinhamento de conceitos;

(d) Camada Conceitual: Nesta camada estão representados os esquemas conceituais e ontologias utilizadas pelo *framework*. Neste trabalho, modelo conceitual trata de uma modelagem sobre um determinado domínio, que serve de base para a modelagem lógica para um banco de dados que armazena dados sobre o domínio. Alguns autores classificam este tipo de modelagem conceitual como Ontologia de peso leve (LASSILA; MCGUINNESS, 2001) (NOY; MCGUINNESS, 2001). Ontologias de contexto utilizadas neste trabalho são consideradas ontologias peso pesado (LASSILA; MCGUINNESS, 2001).

Para realizar a consulta contextualizada de informações, o *framework* utiliza três conjuntos de definições distintas: (i) Uma ontologia que descreve o contexto do ambiente, baseada em outra ontologia independente de aplicação (mais detalhes são apresentados na Seção 5.2), (ii) Um esquema conceitual que descreve o banco de dados de domínio. Este esquema de banco de dados é convertido posteriormente para uma representação em OWL-DL, e (iii) Uma ontologia que define conceitos genéricos e relacionamentos para permitir o alinhamento entre informações de contexto e de domínio;

(e) Persistência: O esquema conceitual e ontologias utilizadas pelo *framework* proposto são persistidas em um banco de dados relacional. A escolha da utilização apenas do banco de dados relacional foi feita devido as seguintes características:

a. A utilização exclusiva de bancos de dados relacionais no *framework* evita problemas relacionados ao *language cacophony problem*[13];
b. Há uma série de ferramentas *opensource* de persistência e recuperação de ontologias em bancos de dados relacionais (FORTUNA et al., 2012). Há ainda a possibilidade de persistência da serialização de ontologias em formatos XML ou JSON-LD utilizando SGBDs relacionais;

O *framework* descrito nesta tese utiliza uma abordagem híbrida para a realização de consultas de dados contextualizados. Pode-se afirmar que o *framework* é híbrido pois utiliza o

[12] O processo de extensão é apresentado na seção 5.5.

[13] *"A utilização de diversas linguagens torna o desenvolvimento mais complexo e difícil se comparado ao mesmo processo utilizando uma linguagem"*. Tradução livre (FOWLER, 2010).

alinhamento de ontologias para integrar conceitos relacionados ao domínio de aplicação e conceitos relacionados a contexto, e realiza um pré-processamento de extensão de consultas composto por algoritmos.

5.2 Modelagem Conceitual

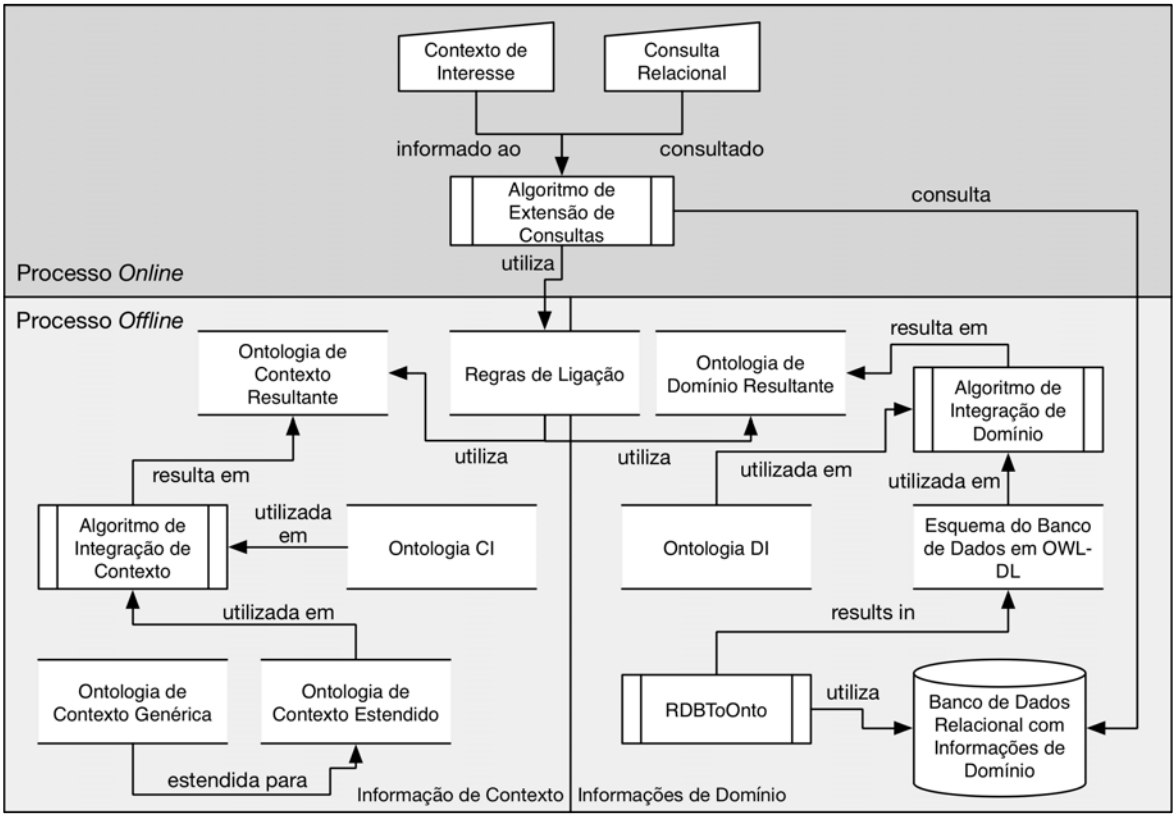
A modelagem de como informações de contexto e informações de domínio são representadas é essencial para realizar a integração das mesmas e para a definição de um *framework* de consulta baseada no alinhamento de conceitos. O *framework* apresentado nesta tese é restrito as seguintes características em relação à modelagem de informações:

- Informações de contexto devem ser representadas em ontologias, pois desta forma contemplam as principais características necessárias para a representação das mesmas em sistemas ubíquos (BETTINI et al., 2010; RIBONI e BETTINI, 2012; STRANG; POPIEN, 2004);
- Informações específicas de domínio são frequentemente representadas de forma relacional e são armazenadas em bancos de dados relacionais para realizar o gerenciamento de informações (BOLCHINI et al., 2013). Além disso, os SGBDs relacionais continuam sendo os mais utilizados em sistemas de informação (DBENGINES, 2015);
- As consultas sobre informações persistidas em bancos de dados relacionais que descrevem o domínio da aplicação são realizados utilizando a linguagem SQL.

A Figura 5.3 apresenta os componentes do *framework* desenvolvido nesta tese para a realização de consultas contextualizadas em bancos de dados relacionais. O *framework* é composto por ontologias, regras de ligação e um conjunto de algoritmos. Estes artefatos em conjunto permitem que consultas realizadas na linguagem SQL utilizem definições de contexto modeladas em ontologias para auxiliar o processo de filtragem de informações.

Para realizar uma consulta dita contextualizada - que utiliza informações de contexto no processo de filtragem de informação; um ***Algoritmo de Extensão de Consultas*** recebe como entradas: (i) uma consulta relacional, descrita na linguagem SQL, e (ii) um contexto de interesse, descrito em um conjunto de axiomas na linguagem OWL-DL. A partir destas entradas, o algoritmo usa as definições pré-definidas da ontologia de contexto, da ontologia que descreve o domínio, e das regras de ligação, para realizar a extensão da consulta SQL, consulta o banco de dados relacional e retorna um conjunto de dados contextualizados. Este processo de extensão é executado para cada consulta que é realizada utilizando o *framework*.

Figura 5.3 – Componentes do *Framework*.



Fonte: Autoria Própria.

Para que as consultas possam ser contextualizadas, o *framework* prevê que três processos devem ser realizados previamente na etapa de modelagem: um deles relacionado à modelagem de contexto, outro deles relacionado à modelagem de domínio e outro relacionado à criação de regras de ligacão.

Em relação ao contexto, prevê-se que uma ***Ontologia Genérica de Contexto*** é utilizada por um sistema ubíquo. Esta ontologia é estendida por projetistas para representar informações de contexto e contextos de interesse necessários para a utilização em consultas contextualizadas, se necessário.

Após a realização desta extensão, a ***Ontologia de Contexto Estendido***[14] é utilizada em um ***Algoritmo de Integração de Contexto***. Este algoritmo integra a ontologia de contexto estendido e os conceitos definidos na ***Ontologia CI*** (*Context Integration*)[15]. O algoritmo tem como saída uma ***Ontologia de Contexto Resultante***, que por sua vez, é utilizada na extensão de consultas e na definição de regras de ligacão.

[14] Ontologia gerada através do *Context Workflow*. Este *workflow* é apresentado posteriormente na seção 5.5.1.

[15] Esta ontologia é apresentada na seção 5.2.1.

Em relação ao domínio, um ***Banco de Dados Relacional*** previamente modelado e implementado é utilizado no *framework* como fonte de informação relacionada ao domínio. Para utilizá-lo no *framework,* é realizada uma extração do esquema do banco de dados no formato OWL-DL.

Para realizar esta extração foi definida a utilização da ferramenta ***RDBToOnto***[16], que implementa o padrão R2RML[17] para a conversão de esquemas relacionais no formato RDF/OWL. Como resultado do processo, uma ***Representação do Esquema no Formato OWL*** é gerada. Esta representação é utilizada em uma entrada, em conjunto com a ***Ontologia DI***[18] (*Domain Integration*) no ***Algoritmo de Integração de Domínio***.

Este algoritmo integra os conceitos provenientes das duas ontologias de entrada e resulta em uma ***Ontologia de Domínio Resultante***, que é utilizado posteriormente no processo de extensão de consultas e na criação de regras de ligação. As relações entre as ontologias, criadas pelos algoritmos de integração, seguem os padrões definidos para a construção de redes de ontologias (DIAZ et al., 2011).

5.2.1 Modelagem de Contexto

Informações de contexto são um conhecimento medido e inferido a partir de fluxos de informação provenientes de ambientes ubíquos (HENRICKSEN, 2003) (STERRIT et al., 2005). Além disso, contexto pode existir independentemente da ação de usuários em um determinado sistema (STRASSNER; O'SULLIVAN, 2009). Assim, a mudança de contexto não é necessariamente realizada a partir de uma mudança feita pelo usuário. Ela pode ser feita por um sistema ou mudar de acordo com uma regra automaticamente inferida (STERRIT et al., 2005) (STRASSNER; O'SULLIVAN, 2009).

De acordo com pesquisas recentes (BETTINI et al., 2010; RIBONI e BETTINI, 2012; MAKRIS et al., 2013), a representação de contexto em ontologias atende aos principais requisitos necessários para a representação de contexto. Para a utilização de contextos, diversas ontologias foram propostas com o objetivo de fornecer definições genéricas de contexto que pudessem ser estendidas de acordo com as necessidades do domínio de aplicação. A ontologia PiVOn (HERVÁS et al., 2011) foi escolhida como ontologia genérica

[16] https://sourceforge.net/projects/rdbtoonto/

[17] https://www.w3.org/TR/r2rml/

[18] Esta ontologia é apresentada na seção 5.2.2.

de contexto. A escolha desta ontologia para o *framework* se deve principalmente pelos seguintes fatores:

(i) Oferece alta representatividade de atividades humanas se comparada a outras ontologias propostas para a representação de contexto (RODRÍGUEZ et al., 2014);

(ii) Oferece exemplos de regras na linguagem SWRL para eliminação de ambiguidades e incremento da qualidade da informação;

(iii) Foi desenvolvida sob a metodologia de redes de ontologias, sendo composta por quatro ontologias independentes (usuários, ambiente, dispositivos e serviços). Desta forma, a ontologia pode ser estendida em partes independentes, de acordo com as necessidades do domínio.

Para utilizar as definições de contexto no *framework* proposto nesta tese, deve-se considerar que uma ontologia de contexto O^C (Definição 1) é composta por definições de classes (C^C), atributos (AT^C), relações (RL^C), indivíduos (I^C), regras de restrição (R^C), regras de inferência (RU^C) definidas em lógica de primeira ordem e axiomas (A^C).

$$O^C = \langle C^C | AT^C | RL^C | I^C | R^C | RU^C | A^C \rangle \quad (1)$$

Esta ontologia de contexto pode ser estendida se houver necessidade de representar contextos específicos para a contextualização de informações. Para integrar a ontologia de contexto no *framework*, foi definida a Ontologia CI (*Context Integration*).

Esta ontologia é composta de conceitos e relações genéricas que descrevem elementos de contexto e contextos de interesse. Através destas definições, é possível que outras ontologias de representação de contexto sejam integradas ao *framework*.

Nesta ontologia, a classe *ContextEntity* foi definida como uma superclasse de elementos de contexto (Axioma 2) e tem como subclasses as classes de elementos de contexto definidas na ontologia PiVOn (Axioma 3).

$$CIOntology{:}\, ContextEntity \sqsubseteq C^C \sqcup R^C \quad (2)$$

$$CIOntology{:}\, ContextEntity \equiv \{PiVOn{:}\, Entity \sqcup PiVOn{:}\, Device \sqcup PiVOn{:}\, User \sqcup PiVOn{:}\, Service \sqcup PiVOn{:}\, Environment\} \quad (3)$$

Contextos de interesse da aplicação são representados pela classe *ContextOfInterest* (Axioma 4). Contextos de interesse são compostos por elementos de contexto e suas subclasses.

$$CIOntology{:}\,ContextOfInterest \equiv \forall CIOntology{:}\,composed\_by^{+}.\,CIOntology{:}\,ContextElement \quad (4)$$

Contextos de interesse podem estar relacionados em si. Um contexto de interesse está relacionado a outro contexto de interesse quando estes contextos possuem elementos de contexto em comum (Axioma 5).

$$\{ContextEntity(?\,ce) \wedge ContextEntity\ (?\,cee) \wedge ContextOfInterest(?\,ci) \wedge ContextOfInterest(?\,cii)\ \wedge composed\_by(?\,ci?\,, ce) \wedge ce\_related\_to(?\,ce, ?\,cee) \rightarrow\ related\_to(?\,ci, ?\,cii)\} \sqsubseteq CIOntology{:}\,RU^{C} \quad (5)$$

Para realizar a integração de uma ontologia previamente definida e utilizada por arquiteturas ubíquas para representação de contexto[19] e a ontologia CI com o *framework*, prevê-se a utilização do algoritmo de integração de contexto descrito a seguir (Figura 5.4).

Figura 5.4 – Algoritmo de integração entre a ontologia de contexto e a ontologia CI.

Algoritmo 1: Integra a ontologia de Contexto e a Ontologia CI

Input: ontologia de contexto $O_c = \{C_c, AT_c, RL_c, I_c, R_c, RU_c, A_c\}$, baseada na ontologia PiVOn, ontologia CI $O_{ci} = \{C_{ci}, AT_{ci}, RL_{ci}, I_{ci}, R_{ci}, RU_{ci}, A_{ci}\}$

Output: ontologia de contexto $O_{cm} = \{C_{cm}, AT_{cm}, RL_{cm}, I_{cm}, R_{cm}, RU_{cm}, A_{cm}\}$, que será integrada no modelo

1 $O_{cm} \leftarrow C_{cm} \leftarrow AT_{cm} \leftarrow RL_{cm} \leftarrow I_{cm} \leftarrow R_{cm} \leftarrow RU_{cm} \leftarrow A_{cm} \leftarrow \emptyset$
2 **foreach** $axioma \in \{C_{ci} \cup AT_{ci} \cup RL_{ci} \cup I_{ci} \cup R_{ci} \cup RU_{ci} \cup A_{ci}\}$ **do**
3 adiciona $axioma$ em C_{cm}
4 **foreach** $contextClass \in C_c$ **do**
5 **if** $contextClass \in \{Entity \cup Device \cup User \cup Service \cup Environment\}$ **then**
6 axioma_aux ← cria axioma $contextClass$ sub_class_of $ContextEntity$
7 adiciona $axioma\_aux$ em C_{cm}
8 axioma_aux ← $UserSituation$ equivalent_to $ContextOfInterest$
9 adiciona $axioma\_aux$ em C_{cm}
10 $AT_{cm} \leftarrow AT_c$
11 $RL_{cm} \leftarrow RL_c$
12 $I_{cm} \leftarrow I_c$
13 $R_{cm} \leftarrow R_c$
14 $RU_{cm} \leftarrow RU_c$
15 $A_{cm} \leftarrow A_c$
16 $O_{cm} \leftarrow \{C_{cm}, AT_{cm}, RL_{cm}, I_{cm}, R_{cm}, RU_{cm}, A_{cm}\}$
17 **return** O_{cm}

Fonte: Autoria Própria.

[19] O algoritmo trata ontologias de representação de contexto baseadas na ontologia PiVOn.

O algoritmo seta as classes mais genéricas da ontologia PiVOn como subclasses da classe *ContextEntity*. Assim, todas as definições de contexto anteriores podem ser utilizadas na definição de consultas contextualizadas. Após a integração das formalizações definidas neste trabalho na ontologia que descreve o contexto gerenciado por arquiteturas ubíquas, as definições são passíveis de serem utilizadas no processo de contextualização de consultas.

5.2.2 Modelagem de Domínio

Dados de contexto também podem ser considerados dados de domínio. Trabalhos recentes (VAVLIAKIS et al., 2013) (BOLCHINI et al., 2013) afirmam que esta é uma questão filosófica, e que escolher quais informações de domínio devem ser consideradas como de contexto deve ser uma decisão de projeto. Neste trabalho foi adotada a mesma postura, ou seja, as informações que são consideradas de contexto e que interferem diretamente no processo de recuperação de informação são definidas por especialistas no domínio da aplicação.

Para realizar a integração entre informações de contexto e informações de domínio, é necessário que o domínio em questão seja representado em um modelo compatível com o modelo utilizado para a representação de contexto. Como informações de contexto são representadas utilizando ontologias[20] faz-se necessária a criação de uma ligação entre as informações representadas em bancos de dados relacionais e as ontologias definidas para a representação de contexto e alinhamento entre contexto e domínio de aplicação.

Para realizar a integração do banco de dados relacional com o *framework* apresentado na tese, faz-se necessária a extração da representação do esquema de banco de dados para um formato compatível com as ontologias descritas no formato OWL-DL. Para realizar a extração na linguagem OWL-DL do esquema de banco de dados, foi utilizada a ferramenta RDBToOnto (LACLAVIK, 2007).

O processo de conversão utilizando a ferramenta RDBToOnto pode ser customizado, conforme as necessidades de cada aplicação. Para a integração feita pelo *framework*, as seguintes características foram adotadas:

- Cada tabela do banco de dados é representada como uma classe;
- Cada atributo de tabela é representado como uma propriedade de dado (*data property*) funcional. O domínio (*domain*) da propriedade de dado é a classe que representa a

[20] No escopo desta tese utiliza-se a linguagem OWL-DL para a representação de ontologias.

tabela do atributo e a imagem (*range*) é o tipo do dado no vocabulário XSD[21]. Chaves primárias seguem a mesma regra de tradução, porém com o nome da tabela adicionado ao nome da propriedade de dado;

• Cada relacionamento 1:N é traduzido como uma propriedade de objeto (*object property*). O domínio (*domain*) da propriedade de objeto é a classe que representa a tabela com cardinalidade 1 e a imagem (*range*) é a classe que representa a tabela com cardinalidade N. Para cada propriedade de objeto criada, é definida também uma nova propriedade de objeto que representa a relação inversa funcional;

Após a realização da extração da representação OWL do esquema de banco de dados, um algoritmo realiza a integração desta ontologia e da ontologia DI (*Domain Integration*). Este algoritmo resulta em uma nova ontologia, que por sua vez é utilizada pelo algoritmo de extensão de consultas. A ontologia DI (O^D) (Definição 6) foi definida como um conjunto de classes (C^D), atributos (AT^D) e relacionamentos (RL^D).

$$O^D = \langle C^D | AT^D | RL^D \rangle \tag{6}$$

A classe *DerivedKnowledge* (Axioma 7) representa qualquer conceito que esteja relacionado ao domínio de aplicação e que possa ser integrado ao *framework*. A classe *ContextualizedQuery* (Axioma 8) representa uma consulta contextualizada.

$$DIOntology: DerivedKnowledge \sqsubseteq C^D \tag{7}$$

$$DIOntology: ContextualizedQuery \sqsubseteq C^D \tag{8}$$

Consultas contextualizadas (*ContextualizedQuery*) (Axioma 9) são compostas por indivíduos da classe *DerivedKnowledge* e relacionados a um conjunto de entidades de contexto, que formam um contexto de interesse e estão relacionados com a consulta.

$$\begin{aligned} &DIOntology: ContextualizedQuery \\ &\equiv \exists composed\_by^{+}.DIOntology: DerivedKnowledge \\ &\sqcap \ \exists related\_to^{+}.(CIOntology: ContextEntity \\ &\sqcup CIOntology: ContextOfInterest) \end{aligned} \tag{9}$$

Consultas contextualizadas podem ser classificadas em três tipos (Axioma 10): (i) Domínio como contexto (*DomainAsContext*), (ii) Domínio para elemento de contexto (*DomainForContextElement*) e (iii) Domínio para valor de contexto (*DomainForContextValue*). Cada um destes tipos está relacionado a uma regra de ligação (a definição de cada regra é apresentada na seção 5.2.3).

[21] https://www.w3.org/TR/xmlschema-2/

$$(DIOntology{:}\,DomainAsContext \sqcup DIOntology{:}\,DomainForContextElement \sqcup DIOntology{:}\,DomainForContextValue) \sqsubseteq DIOntology{:}\,ContextualizedQuery \quad (10)$$

Para realizar a integração de uma ontologia previamente definida através do processo de conversão do esquema de banco de dados para uma representação no formato OWL com o *framework*, prevê-se a utilização de um algoritmo, descrito a seguir (Figura 5.5).

Figura 5.5 – Algoritmo de integração entre a representação do banco de dados no formato OWL-DL e a ontologia DI.

Algoritmo 2: Integra a ontologia de domínio e a ontologia DI

Input: ontologia de domínio $O_D = \{C_D, AT_D, RL_D\}$, gerada pela ferramenta RDBToOnto

Output: ontologia de domínio $O_{Dm} = \{C_{Dm}, AT_{Dm}, RL_{Dm}, RU_{Dm}, A_{Dm}\}$, que será utilizada no modelo

1 $O_{Dm} \leftarrow C_Dm \leftarrow AT_{Dm} \leftarrow RL_{Dm} \leftarrow RU_{Dm} \leftarrow A_{Dm} \leftarrow \emptyset$
2 **foreach** $domainClass \in C_D$ **do**
3 axioma_aux ← $domainClass$ sub_class_of $DerivedKnowledge$
4 insere $axioma\_aux$ em C_{Dm}
5 $AT_{Dm} \leftarrow AT_D$
6 $RL_{Dm} \leftarrow RL_D$
7 $A_{Dm} \leftarrow A_D$
8 $O_{Dm} \leftarrow \{C_{Dm}, AT_{Dm}, RL_{Dm}, RU_{Dm}, A_{Dm}\}$
9 **return** O_{Dm}

Fonte: Autoria Própria.

O algoritmo define todas as classes resultantes do processo de extração do esquema de banco de dados como subclasses da classe *DerivedKnowledge*. A saída do algoritmo de integração é uma representação no formato OWL-DL que permite com que os conceitos pré-definidos no banco de dados sejam utilizados na criação de regras de ligação que serão utilizadas pelo algoritmo de extensão de consultas.

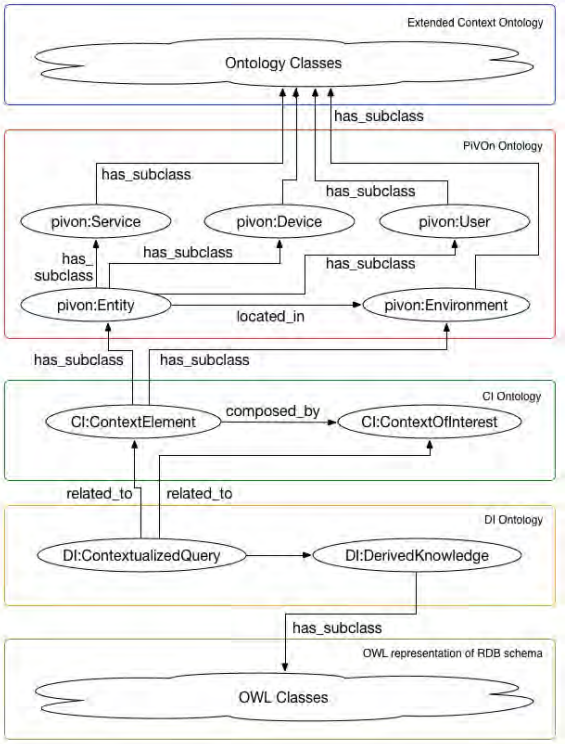
A Figura 5.6 apresenta a rede de ontologias (apenas as classes utilizadas no processo de integração) gerada após a definição das etapas de (i) definição da ontologia de contexto, (ii) integração com a ontologia CI, (iii) extração do esquema de banco de dados no formato OWL-DL e (iv) integração com a ontologia DI.

Após a definição da rede de ontologias, os projetistas utilizam os conceitos e relações para a definição de regras de ligação. A expansão de consultas ocorre em duas fases diferentes. Na primeira fase, um especialista no domínio da aplicação e um engenheiro de ontologias configuram manualmente um grupo de regras de ligacão.

Na segunda fase, um algoritmo interpreta as regras de ligacão criadas e filtra os resultados de uma consulta, dada uma expressão descrita em álgebra relacional. O processo de

criação das regras de ligacão, bem como a forma de representação é apresentado na próxima seção.

Figura 5.6 – Rede de ontologias gerada após a realização dos processos de integração.



Fonte: Autoria Própria.

5.2.3 Regras de Ligacão

Regras de ligacão permitem definir modos de ligação entre informações representadas nas ontologias de contexto e de domínio e as relações do banco de dados. Estas regras são interpretadas pelo algoritmo de extensão de consultas e traduzidas para regras de filtragem. Cada regra é representada em um arquivo JSON que é armazenado na mesma instância de banco de dados de domínio da aplicação[22].

5.2.3.1 Regra *Domínio e Contexto*

A primeira regra de ligacão definida foi a regra ***Domínio E Contexto*** (↔). A regra é representada através da classe $DIOntology{:}\,DomainAsContext$.

Definição: Seja *D* um esquema relacional, *R* uma tabela (relação) que pertence ao esquema *D*, *t* uma tupla de *R* e *C* uma classe de entidade de contexto, onde $C \sqsubseteq$

[22] Mais informações sobre a estrutura das representações em JSON é apresentada na seção 5.3.

$\{CIOntology{:}\,ContextEntity \sqcup CIOntology{:}\,ContextofInterest\}$, a regra de ligação $R_{(t)} \leftrightarrow C$ implica na criação de um novo indivíduo da classe *C* que possui uma propriedade de dado com o mesmo nome da chave primária de *t* e com o mesmo valor. Esta criação é necessária porque a ontologia que descreve o domínio da aplicação descreve apenas o esquema do banco de dados, sem a representação de cada dado armazenado no banco de dados.

Além disso, o algoritmo realiza a criação de um indivíduo da classe $DIOntology{:}\,DomainAsContext$, que representa a regra de ligação na rede de ontologias. Após a criação deste indivíduo, o mesmo é relacionado através do relacionamento *composed_by* com a classe que representa a tabela *R*, e através do relacionamento *related_to* com a classe de contexto *C*. O processo de modificação das ontologias DI e CI para representar a regra é apresentado a seguir, na Figura 5.7.

Figura 5.7 – Algoritmo de criação da regra *DomínioEContexto* (parte 1).

Algoritmo 3: PROCESSO DE ASSOCIAÇÃO $R_{(t)} \leftrightarrow C$ (PARTE 1)

Input: tupla *t*, tabela *R*, classe de contexto *C*, ontologia DI, ontologia CI
Output: modificações nas ontologias DI e CI para associação $R_{(t)} \leftrightarrow C$

```
1  class_aux ← obtemClasseReferenteATupla(t, ontologia DI)
   /* cria indivíduo na ontologia DI                                  */
2  i_d ← criaNovoIndividuo(ontologia DI)
3  i_d.type_of ← class_aux
4  i_d.uid ← nome de class_aux + valor da chave primária de t
5  i_d_data_property ← nova propriedade de dado na ontologia DI
6  i_d_data_property.uid ← nome da chave primária de t
7  i_d_data_property.domain ← class_aux
8  i_d_data_property.range ← tipo xsl da chave primária de t
9  i_d_data_property.value ← valor da chave primária de t
10 insereNaOntologia(i_d, ontologia DI)
11 insereNaOntologia(i_d_data_property, ontologia DI)
   /* cria indivíduo na ontologia CI                                  */
12 i_d2 ← criaNovoIndividuo(ontologia CI)
13 i_d2.type_of ← C
14 i_d2.uid ← nome de class_aux + valor da chave primária de t + '_'
15 insereNaOntologia(i_d2, ontologia CI)
   /* cria indivíduo do tipo DomainAsContext e associa aos indivíduos de CI e
      DI                                                              */
16 i_d3 ← criaNovoIndividuo(ontologia DI)
17 i_d3.type_of ← DomainAsContext
18 i_d3.uid ← DomainAsContext + '_' + i_d.uid
19 i_d_obj_property ← nova propriedade de objeto na ontologia DI
20 i_d_obj_property.uid ← composed_by
21 i_d_obj_property.domain ← i_d3
22 i_d_obj_property.range ← i_d
23 i_d_obj_property2 ← nova propriedade de objeto na ontologia DI
24 i_d_obj_property2.uid ← related_to
25 i_d_obj_property2.domain ← i_d3
26 i_d_obj_property2.range ← i_d2
27 insereNaOntologia(i_d3, ontologia DI)
28 insereNaOntologia(i_d_obj_property, ontologia DI)
29 insereNaOntologia(i_d_obj_property2, ontologia DI)
30 return ontologias DI e CI modificadas
```

Fonte: Autoria Própria.

O processo de criação da regra que utiliza o algoritmo anterior é apresentado a seguir, na Figura 5.8. Além das modificações na ontologia, uma definição em JSON é criada para representar a regra. Esta representação em JSON é utilizada pelo algoritmo extensor de consultas (mais informações sobre a estrutura das representações em JSON é apresentada na seção 5.3).

Figura 5.8 – Algoritmo de criação da regra *DomínioEContexto* (parte 2).

Algoritmo 4: PROCESSO DE ASSOCIAÇÃO $R_{(t)} \leftrightarrow C$ (PARTE 2)

Input: tupla t, tabela R, classe de contexto C, ontologia DI, ontologia CI
Output: a associação $R_{(t)} \leftrightarrow C$ representada nas modificações realizadas nas ontologias DI e CI, e no arquivo rule_in_json

```
1  class_aux ← obtemClasseReferenteATupla(t, ontologia DI)
   /* se t é uma tupla específica                                   */
2  if t ≠ NULL then
3      executaAlgoritmo3(t, R, C, ontologia DI, ontologia CI)
4      achou ← 0
5      foreach rule_in_json ∈ association_rules do
6          if class_aux = rule_in_json.domain then
7              rule_in_json2 ← rule_in_json
8              insere i_d_data_property.value em rule_in_json.values
9              achou ← 1
   /* se t for null, cria uma representação para cada tupla da tabela R  */
10 else
11     foreach tupla ∈ R do
12         executaAlgoritmo3(tupla, R, C, ontologia DI, ontologia CI)
13 if achou = 0 then
14     rule_in_json2 ← novoarquivoJSON
15     insere type : domainAsContext em rule_in_json
16     insere domain : class_aux.uid em rule_in_json
17     insere context : C.uid em rule_in_json
18     insere key : i_d_data_property.uid em rule_in_json
19     insere values:[i_d_data_property.value] em rule_in_json
20 return rule_in_json2
```

Fonte: Autoria Própria.

Como exemplo de utilização da regra de ligacão, vamos considerar a tabela *module* e os dados armazenados nesta relação, apresentados na Tabela 5.1. A tabela representa um conjunto de módulos de um curso.

Tabela 5.1 – Informações armazenadas na tabela *module*.

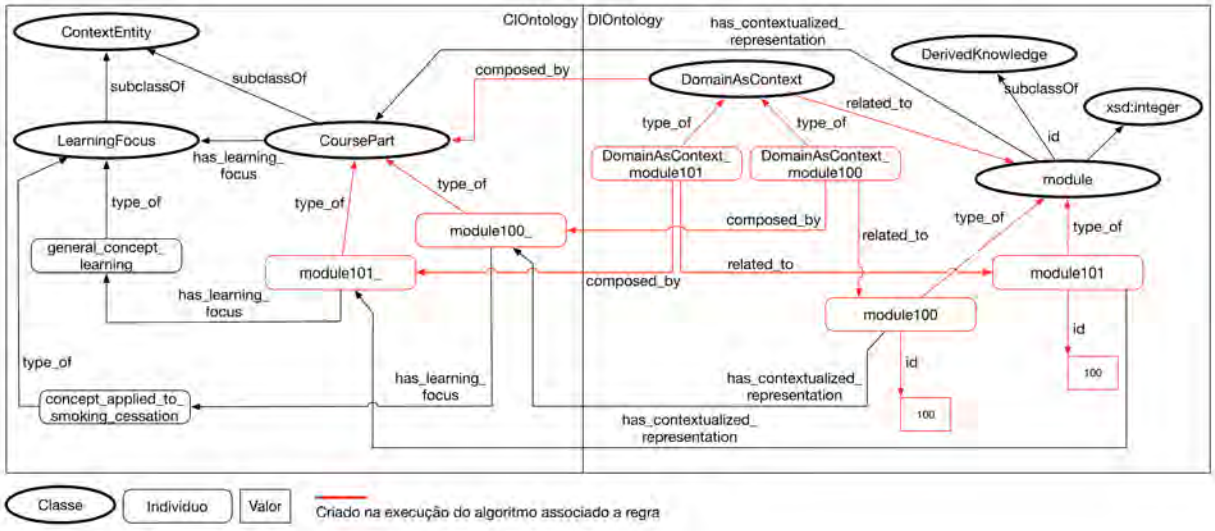
tupla	***id***	***display_name***	***graded***	***parent***	***format***	***type***
1	100	Caso Clínico 1	1	NULL	html	main
2	101	Caso Clínico 2	1	NULL	html	main
3	102	Caso Clínico 3	0	1	html	opt
4	103	Caso Clínico 3	0	1	html	opt

Fonte: Autoria Própria.

Cada módulo do curso é identificado através de um atributo *id*, possui um *display_name*, que representa o nome do módulo, um atributo *graded*, que representa se é obrigatório completar o módulo para completar o curso, um atributo *parent*, que representa se o módulo é um submódulo de outro, mais genérico, um atributo *format*, que representa o formato no qual o módulo é apresentado e um atributo *type*, que representa se o módulo é do tipo principal (*main*) ou se é de outro tipo.

Para a integração entre contexto e domínio, os projetistas necessitam associar novas informações a cada módulo, que não foram previamente modeladas na relação. Neste exemplo, vamos considerar que cada módulo deve possuir, além das informações apresentadas, um dado que indica sobre qual conceito específico o módulo trata. Assim, os projetistas modelaram previamente uma classe *CoursePart* na ontologia de contexto, que está relacionada à classe *LearningFocus*. Assim, a definição das regras de ligação **module$_{(tupla1)}$** ↔ ***CoursePart*** e **module$_{(tupla2)}$** ↔ ***CoursePart*** resulta nas classes e indivíduos apresentados na Figura 5.9.

Figura 5.9 – Exemplo de ligacão entre o conceito de domínio *module* e o elemento de contexto *ContextPart*.



Fonte: Autoria Própria.

Após a definição da regra de ligação, foi possível associar informações de contexto aos indivíduos que representam informações de domínio. Neste exemplo, o módulo *module101_* foi associado ao conceito *general_concept_learning*, e o módulo *module100_* foi associado ao conceito *concept_applied_to_smoking_cessation*, indicando o foco de aprendizado de cada um dos módulos.

5.2.3.2 Regra *Domínio Para um Valor Específico de Contexto*

A regra **Domínio Para um Valor Específico de Contexto** (⇝) foi criada para relacionar uma expressão de álgebra relacional a um valor específico de uma propriedade de dado de uma subclasse de *ContextEntity*.

Definição: Seja $\mathcal{D}$ um esquema relacional, $rae(\mathcal{D})$ uma expressão de álgebra relacional sobre $\mathcal{D}$, $\mathcal{C}$ uma classe de entidade contexto na ontologia CI, *ci_p* uma propriedade de dados de $\mathcal{C}$, val um valor específico para *ci_p*, *di_p* um atributo de domínio, onde *di_p* ∈ $\{rae(\mathcal{D}) \cup NULL\}$, a expressão $\mathcal{C}_{(ci\_p,val,di\_p)} \rightsquigarrow rae(\mathcal{D})$ equivale a σ (*di_p* = (val)) ($rae(\mathcal{D})$).

A regra é representada na ontologia DI através da classe $DIOntology{:}\,DomainForContextValue$ (Axioma 11). Cada indivíduo da classe $DomainForContextValue$ possui uma propriedade de dado *rae*, do tipo *xsd:string*, que representa a expressão em álgebra relacional associada à regra de ligação, uma propriedade de dado *ci_p*, do tipo *xsd:string*, que representa a propriedade de contexto utilizada na regra, uma propriedade de dado *di_p*, do tipo *xsd:string*, que representa a propriedade de domínio utilizada na regra e uma propriedade de dado *val*, que descreve o valor da propriedade de contexto utilizada na regra, do tipo *xsd:string*.

$$DIOntology{:}\,DomainForContextValue$$
$$\equiv\ exacly\ 1\ ci_p.xsd{:}\,string\ \sqcap\ exacly\ 1\ val.xsd{:}\,string \sqcap\ \exists di\_p.xsd{:}\,string$$
$$\sqcap\ exacly\ 1\ rae.xsd{:}\,string \qquad (11)$$

O processo de criação da regra é apresentado a seguir, na Figura 5.10.

Como exemplo de utilização, considerando o uso da tabela *module*, apresentada anteriormente na Tabela 5.1, e o *contexto_de_interesse_1*, apresentado na Figura 5.11. Este contexto de interesse representa um aluno que acessa um MOOC através de um dispositivo móvel.

Considerando que um modelador desejasse especificar que somente módulos no formato HTML pudessem ser apresentados aos usuários que utilizam dispositivos do tipo *limited_viewing*, ou seja, que possuem capacidades limitadas de apresentação, ele deve criar a regra $Device_{(type,limited\_viewing,NULL)} \rightsquigarrow \big(\sigma(format =' html')(module)\big)$.

Figura 5.10 – Algoritmo de definição de regra Domínio para um valor específico de contexto.

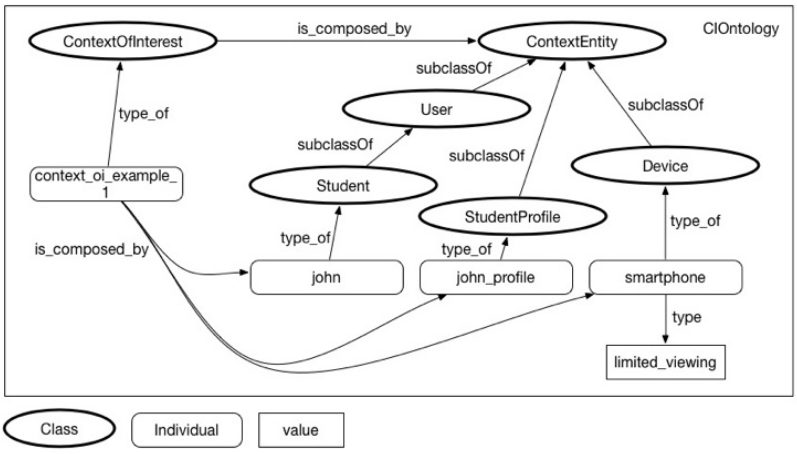
Algoritmo 5: PROCESSO DE ASSOCIAÇÃO $C_{(ci\_p,val,di\_p)} \rightsquigarrow rae(D)$

Input: classe de contexto C, propriedade de contexto ci_p, valor val, atributo de domínio di_p, ontologia DI, ontologia CI
Output: a associacão $C_{(ci\_p,val,di\_p)} \rightsquigarrow rae(D)$

```
/* cria indivíduo do tipo DomainForContextValue                    */
i_d ← criaNovoIndividuo(ontologia DI)
i_d.type_of ← DomainForContextValue
i_d.uid ← DomainForContextValue + '_' + i_d.uid
i_d_data_property ← nova propriedade de dado na ontologia DI
i_d_data_property.uid ← rae
i_d_data_property.domain ← i_d
i_d_data_property2.range ← rae(D)
insereNaOntologia(i_d, ontologia DI)
insereNaOntologia(i_d_data_property, ontologia DI)
i_d_data_property ← nova propriedade de dado na ontologia DI
i_d_data_property.uid ← ci_p
i_d_data_property.domain ← i_d
i_d_data_property2.range ← ci_p
insereNaOntologia(i_d_data_property, ontologia DI)
if di_p ≠ NULL then
    i_d_data_property ← nova propriedade de dado na ontologia DI
    i_d_data_property.uid ← di_p
    i_d_data_property.domain ← i_d
    i_d_data_property2.range ← di_p
    insereNaOntologia(i_d_data_property, ontologia DI)
i_d_data_property ← nova propriedade de dado na ontologia DI
i_d_data_property.uid ← val
i_d_data_property.domain ← i_d
i_d_data_property2.range ← val
insereNaOntologia(i_d_data_property, ontologia DI)
/* cria representação em JSON                                      */
rule_in_json ← novoarquivoJSON
rule_in_json ←
  insereNoJSON("type" : "DomainForContextValue","context" :
  C.uid,"context_property" : ci_p,"value" : val)
if di_p ≠ NULL then
    insere "domain" : C.uid em rule_in_json
insere "rae" : rae(D) em rule_in_json
return rule_in_json
```

Fonte: Autoria Própria.

Figura 5.11 – Contexto de interesse que representa um aluno utilizando um dispositivo móvel para acessar um MOOC.



Fonte: Autoria Própria.

5.2.3.3 Regra *Domínio Para Elemento Específico De Contexto*

A regra **Domínio para Elemento de Contexto Específico** (↠) foi criada para relacionar uma expressão de álgebra relacional a um indivíduo na ontologia que descreve um elemento de contexto específico.

Definição: Seja $\mathcal{D}$ um esquema relacional, $rae(\mathcal{D})$ uma expressão de álgebra relacional sobre $\mathcal{D}$, $\mathcal{C}$ uma classe de entidade contexto na ontologia DI, ci_i um indivíduo do tipo $\mathcal{C}$, a expressão $\mathcal{C}_{(ci\_i)} \twoheadrightarrow rae(\mathcal{D})$ define que uma expressão $rae(\mathcal{D})$ é incorporada a consulta quando o indivíduo de contexto ci_i é parte do contexto de interesse informado.

A regra é representada na ontologia DI através da classe $DIOntology{:}\,DomainForContextElement$ (Axioma 12). Cada indivíduo da classe $DomainForContextElement$ deve estar relacionado a somente um indivíduo que representa o elemento de contexto e possui uma propriedade de dado *rae*, do tipo *xsd:string*, que representa a expressão em álgebra relacional associada à regra de ligação.

$$\begin{aligned} & DIOntology{:}\,DomainForContextElement \\ \equiv\ & exacly\ 1\ related\_to^{+}.(CIOntology{:}\,ContextEntity \\ \sqcup\ & CIOntology{:}\,ContextOfInterest) \\ \sqcap\ & exacly\ 1\ rae.xsd{:}\,string \end{aligned} \tag{12}$$

O processo de criação a regra é apresentado na Figura 5.12.

Figura 5.12 – Algoritmo de criação da regra Domínio para Elemento de Contexto Específico.

Algoritmo 6: PROCESSO DE ASSOCIAÇÃO $C_{(ci\_i)} \twoheadrightarrow rae(D)$

Input: classe de contexto C, elemento de contexto ci_i, expressão em álgebra relacional $rae(D)$, ontologia DI
Output: a associação $C_{(ci\_i)} \twoheadrightarrow rae(D)$, *representada nas modificações nas ontologias DI e CI, e no arquivo JSON rule_file*

```
/* cria indivíduo do tipo DomainForContextElement                    */
1  i_d ← criaNovoIndividuo(ontologia DI)
2  i_d.type_of ← DomainForContextElement
3  i_d.uid ← DomainForContextElement + '_' + i_d.uid
4  i_d_obj_property ← nova propriedade de objeto na ontologia DI
5  i_d_obj_property.uid ← related_to
6  i_d_obj_property.domain ← i_d
7  i_d_obj_property.range ← ci_i
8  i_d_data_property ← nova propriedade de dado na ontologia DI
9  i_d_data_property.uid ← rae
10 i_d_data_property.domain ← i_d
11 i_d_data_property2.range ← rae(D)
12 insereNaOntologia(i_d, ontologia DI)
13 insereNaOntologia(i_d_obj_property, ontologia DI)
14 insereNaOntologia(i_d_data_property, ontologia DI)
   /* cria a definição em JSON da regra                                */
15 rule_in_json ← novoarquivoJSON
16 rule_in_json ← insereNoJSON("type" : "DomainForContextElement",
     "context" : C.uid, "context_individual" : ci_i, "rae" : rae(D))
17 return rule_in_json
```

Fonte: Autoria Própria.

Considerando as informações de domínio previamente apresentadas na Tabela 5.1 e as informações de contexto apresentadas na Figura 5.6, vamos imaginar que, por uma decisão de projeto, apenas os módulos do tipo *main* podem ser associados a *smartphones*, e que smartphones são representados na ontologia de contexto como um indivíduo *smartphone*. A associação entre o indivíduo e a filtragem de módulos por tipo pode ser feita com a definição da regra $Device_{(\text{smartphone})} \twoheadrightarrow \sigma(type =' main')(module)$.

5.3 Serialização e Persistência

Para realizar a recuperação de dados baseada em contexto, foi utilizada uma forma de persistência que permite com que dados relacionais e dados de contexto sejam armazenados na mesma instância de banco relacional sem alterar o esquema conceitual previamente definido para o domínio[23]. Desta forma, a consulta de dados e a consulta de contexto podem ser feitas sobre o mesmo esquema pelo *framework*.

Ontologias representadas em OWL-DL podem ser serializadas de diversas formas, entre as mais comuns estão RDF/XML, OWL/XML, Sintaxe Funcional, Sintaxe Manchester e Triplas *Turtle* (W3C, 2015). Ferramentas foram propostas para a persistência de ontologias em bancos de dados, sejam eles relacionais (VYSNIAUSKAS et al., 2006) (DE LABORDA et al., 2005) ou não relacionais (WANG et al., 2012) (DENTLER et al., 2012). Porém, de acordo com estudos recentes (MAARALA et al., 2014) (SU et al., 2015) a linguagem JSON-LD demonstra vantagens importantes sob a representação em RDF, principalmente em relação ao tamanho das mensagens e simplicidade na representação.

Considerando a existência de um esquema relacional relacionado ao domínio de aplicação R^D, o conjunto de relações utilizadas pelo *framework* (R^F) é composto (Definição 13) pelas relações que descrevem o domínio, previamente modeladas, as relações que armazenam as representações em JSON-LD que compõem a ontologia de contexto (R^C) e pela relação que contém as descrições das regras de ligacão (R^L), utilizadas pelo algoritmo extensor de consultas.

$$R^F = \langle R^D | R^C | R^L \rangle \tag{13}$$

[23] Sem alteração nas tabelas que representam o domínio e foram previamente definidas. O *framework* prevê a criação de novas tabelas para o armazenamento das ontologias utilizadas no processo e as regras de ligação.

Cada regra de ligacão modelada é representada em um arquivo JSON, que é armazenado na mesma instância de banco de dados utilizada na aplicação, na tabela *sqleco_link_rules* (Definição 14). A tabela possui dois atributos, um identificador *id*, e um JSON *desc*, que armazena a definição da regra no formato JSON.

$$R^L = sqleco\_link\_rules(\underline{integer\ id}, json\ desc) \quad (14)$$

Para armazenar a rede de ontologias utilizada no *framework*, é utilizada a tabela *sqleco_context* (Definição 15). Esta tabela armazena as representações no formato JSON da rede de ontologias gerada pelo *framework*.

$$R^C = sqleco\_context(\underline{integer\ id}, json\ ontology\_axiom) \quad (15)$$

A escolha desta forma de armazenamento se deve ao fato de que consultas e inferências complexas sobre contexto são realizados pelo *middleware* ubíquo de gerenciamento de contexto.

O *framework* apresentado nesta tese apenas utiliza cópias destas definições e, portanto, não necessita de esquemas complexos de inferência (com suporte a OWL *Full*) ou com suporte à linguagem SPARQL. Desta forma, optou-se pela utilização de uma ferramenta de serialização de ontologias OWL-DL no formato JSON, compatível com uma API para gerenciamento de ontologias na linguagem Java[24]. Assim o *framework* gerencia a rede de ontologias utilizando a API OWL-API[25] e realiza a persistência da ontologia na mesma instância de banco de dados de domínio. Esta escolha de projeto permite também que o algoritmo extensor de consultas utilize apenas a linguagem SQL para recuperar informações de contexto, das regras de associação e do domínio da aplicação.

5.4 Extensão de Consultas

Para realizar a extensão de consultas, são utilizados dois tipos de informação: (i) um contexto de interesse, representado na linguagem OWL-DL e informado por um *middleware* ubíquo e (ii) uma consulta relacional, representada como uma expressão em álgebra relacional[26]. A partir destas duas informações, um algoritmo de extensão realiza a extensão da

[24] http://dx.doi.org/10.5281/zenodo.10561

[25] http://owlapi.sourceforge.net

[26] A implementação da arquitetura que suporta o modelo realiza um mapeamento de expressões em álgebra relacional para expressões na linguagem SQL.

consulta original, realiza a consulta na base de dados e retorna os dados devidamente contextualizados.

Para as definições seguintes, vamos definir $\mathcal{D}$ como um esquema relacional, $\mathcal{R}$ como uma tabela pertencente a $\mathcal{D}$, X como um conjunto de atributos de $\mathcal{R}$, θ um conjunto de condições sobre $X(\mathcal{R})$ e $rae(\mathcal{R})$ uma expressão de álgebra relacional sobre $\mathcal{R}$. Deste modo, uma expressão $rae(\mathcal{R})$ é equivalente a expressão $\pi\ (X)\ (\sigma\ \theta)\ (\mathcal{R})$.

Como foi apresentado anteriormente, regras do tipo **Domínio para Elemento de Contexto Específico** (↠) e **Domínio Para um Valor Específico de Contexto** (⇝) possuem uma expressão de ágebra relacional associada à regra, enquanto a regra ***Domínio E Contexto*** (↔) não possui expressão de álgebra relacional associada.

A extensão de consultas proposta nesta tese utiliza o contexto de interesse informado pelo *middleware* ubíquo como exemplo. A partir deste contexto de interesse, informado no momento da consulta, o algoritmo de extensão de consultas verifica se há alguma expressão de consulta relacionada ao elemento ou valor de contexto e integra esta expressão na consulta original.

Sendo $rae^{orig}(\mathcal{R})$ uma expressão em álgebra relacional informada no momento da consulta, equivalente a $\pi\ (X^{orig})\ (\sigma\ \theta^{orig})\ (\mathcal{R}^{orig})$ e $rae^{assoc}(\mathcal{R})$ uma expressão em álgebra relacional associada a uma regra de ligacão previamente modelada, equivalente a $\pi\ (X^{assoc})\ (\sigma\ \theta^{assoc})\ (\mathcal{R}^{assoc})$, a expressão em álgebra relacional resultante $rae^{res}(\mathcal{R})$ pode ser definida como $\pi\ (X^{orig}\ ,\ X^{assoc})\ (\sigma\ \theta^{orig} \wedge \theta^{assoc})\ (\mathcal{R}^{orig}\ \bowtie \mathcal{R}^{assoc})$.

Como contextos de interesse são compostos por uma série de elementos de contexto, há a possibilidade de que mais de um elemento de contexto possua uma regra de ligação associada. Neste caso, considerando que $rae^{el\_a}(\mathcal{R})$ é uma expressão associada a um elemento de contexto a e $rae^{el\_b}(\mathcal{R})$ é uma expressão associada a um elemento de contexto b, um contexto de interesse que contenha os elementos de contexto a e b resulta em uma expressão $\pi\ (X^{orig}\ ,\ X^{el\_a}, X^{el\_b})\ (\sigma\ \theta^{orig} \wedge (\theta^{el\_a}\ \vee\ \theta^{el\_b}))\ (\mathcal{R}^{orig}\ \bowtie \mathcal{R}^{el\_a} \bowtie\ \mathcal{R}^{el\_b})$.

Como as regras do tipo regra ***Domínio E Contexto*** (↔) não possuem expressões de consulta associadas, elas são tratadas de forma diferente pelo algoritmo. Neste caso, sendo *e* um elemento de contexto presente em um contexto de interesse, onde *e* está associado a uma regra do tipo *Domínio E Contexto*, sendo *identificador* o nome do atributo utilizado como identificador na tabela associada a regra *Domínio E Contexto,* a expressão resultante é definida por $\pi\ (X^{orig})\ (\sigma\ \theta^{orig} \wedge identificador = e.valor\ do\ identificador)\ (\mathcal{R}^{orig})$.

Na Figura 5.13 é apresentado o algoritmo de extensão de consultas proposto nesta tese. Nele, a partir de uma expressão de álgebra relacional que descreve uma consulta originalmente realizada em um banco de dados, e um contexto de interesse, representado por um conjunto de indivíduos de uma ontologia, gera uma consulta resultante, que é realizada posteriormente no banco de dados.

Figura 5.13 – Algoritmo extensor de consultas.

Algoritmo 7: EXTENSÃO DE *rae* BASEADA NO CONTEXTO C_{int}

Input: contexto de interesse C_{int}, expressão de álgebra relacional *rae*
Output: expressão em ágebra relacional resultante rae_r

```
1  rae_r ← ∅
2  rae_r.proj ← rae.proj
3  rae_r.sel ← rae.sel
4  rae_r_aux.sel ← ∅
5  rae_r.relations ← rae.relations
6  result_set ← consulta regras de associação, onde rule."context" ∈
     elementos de contexto de C_int
7  foreach i_cxt ∈ C_int do
8      foreach result ∈ result_set do
9          if i_cxt.uid = result.context then
               /* trata regras Dominio E Contexto                    */
10             if rule.type = DomainAsContext ∧ rule.domain ∈
                rae_r.relations then
11                 dk ← rule.key
12                 cv ← i_cxt.dk.value
13                 rae_r_aux.sel ← rae_r_aux.sel ∨ (dk = cv)
14             else
                   /* trata regras Dominio para Elemento de contexto específico
                      */
15                 if rule.type = DomainForSpecificContextElementValue
                    then
16                     prop_aux ← result.context_property
17                     aux ← i_cxt data property as prop_aux
18                     if aux ≠ NULL then
19                         rae_r_aux.sel ← rae_r_aux.sel ∨ rule.rae.sel
20                         rae_r.relations ← rae_r.relations ⋈
                            rule.rae.relations
                       /* trata regras Dominio para valor de contexto específico
                          */
21                 else
22                     rae_r_aux.sel ← rae_r_aux.sel ∨ rule.rae.sel
23                     rae_r.relations ← rae_r.relations ⋈ rule.rae.relations
24 rae_r.sel ← rae_r_aux.sel ∨ rae_r.sel
25 return rae_r
```

Fonte: Autoria Própria.

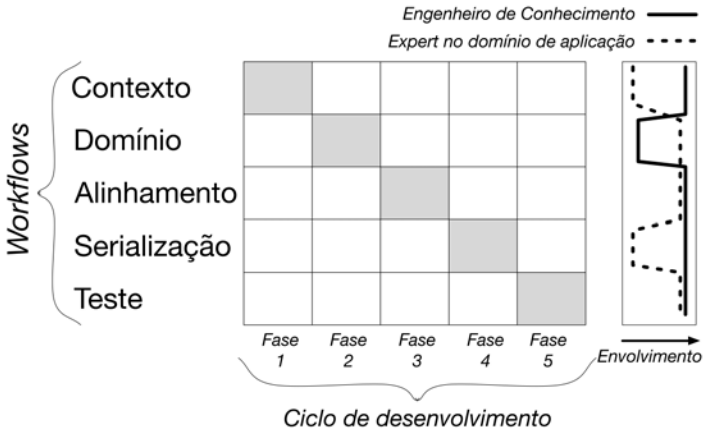
Para realizar a integração das definições propostas nesta tese com consultas pré-existentes e ontologias que descrevem contexto, uma metodologia de utilização do *framework* é proposta e apresentada a seguir.

5.5 Metodologia de Utilização

Para realizar a implementação do *framework* em um sistema em funcionamento, considerando o uso de um banco de dados relacional e uma ontologia de contexto pré-definidos, foi definida uma metodologia de implementação. A metodologia de implementação foi baseada na metodologia UPON (DE NICOLA et al., 2009), que foi definida para a criação de ontologias de larga escala. A metodologia desenvolvida é uma extensão da metodologia UPON e da metodologia de desenvolvimento de software UP (*Unified Process*) (JACOBSON et al., 1999). A metodologia é composta por ciclos de implementação evolutivos e compostos de *workflows* de trabalho que descrevem uma série de operações. O número de ciclos de implementação varia conforme a necessidade dos projetistas em associar: (*i*) novos elementos de contexto na filtragem de informações ou (*ii*) uma nova fonte de informação no processo de filtragem (banco de dados ou conjunto de relações).

Cada *workflow* tem entradas e saídas. Em cada ciclo de desenvolvimento devem ser executados 5 *workflows*: *Context Workflow*, *Domain Workflow*, *Alignment Workflow*, *Serialization Workflow* e *Query Test Workflow*. Cada *workflow* é executado em uma fase de implementação. A visão geral da metodologia é apresentada na Figura 5.14.

Figura 5.14 – Visão geral das etapas que compõem a metodologia de implementação do *framework*.



Fonte: Autoria Própria.

O nível de envolvimento com o suporte na implantação do *framework* pelos *experts* no domínio de aplicação e engenheiros de ontologias varia conforme cada um dos *workflows* que compõem as fases da metodologia. A estrutura de *workflows* para cada fase com entradas e saídas foi baseada na metodologia UPON (DE NICOLA et al., 2009), que é utilizada na fase 1.

5.5.1 *Context Workflow*

O *Context Workflow* é um conjunto de passos definidos para definir ou estender uma ontologia de contexto para que a mesma seja utilizada na extensão de consultas através da aplicação do *framework* proposto nesta tese. Este *workflow* segue os mesmos passos definidos na Metodologia UPON de definição de ontologias, com algumas modificações na ordem de execução dos workflows definidos nesta metodologia. Uma visão geral dos processos do *Context Workflow* é apresentada na Figura 5.15. Os processos marcados na figura foram adicionados na metodologia proposta neste trabalho. Os processos são descritos a seguir:

- **Definir o Domínio de Interesse e Escopo**: Consiste na identificação dos principais conceitos que devem ser representados, suas características e a definição do foco (escopo) do domínio que será representado. Para isto, devem ser realizados compromissos ontológicos[27] no processo. Originalmente, na Metodologia UPON, há outro processo realizado nesta etapa, chamado *Definição do propósito de negócio*. Este processo foi retirado nesta metodologia, pois a metodologia já define previamente que o propósito de definição é a extensão de uma ontologia de contexto pré-existente para a extensão de consultas relacionais;

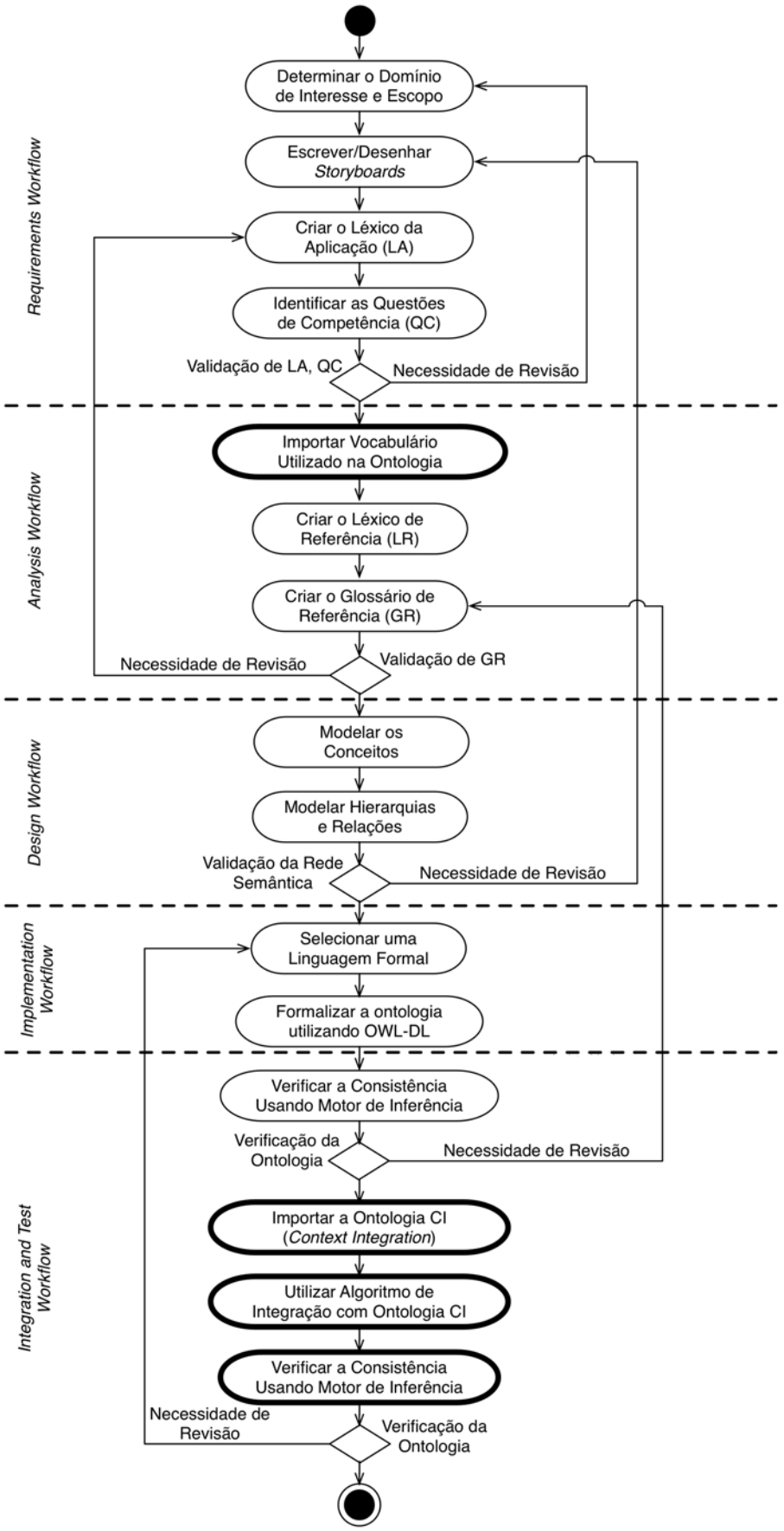
- **Escrever *Storyboards***: Nesta etapa o *expert* no domínio de aplicação descreve um ou uma série de *Storyboards* que descrevem sequencias de atividades de um determinado cenário. Este cenário descreve um conjunto de situações e contextos da aplicação;

- **Criar o léxico da aplicação (LA)**: O léxico da aplicação é um conjunto dos principais termos, coletados a partir das *Storyboards* desenvolvidas no processo anterior, de documentos provenientes da aplicação e do auxílio do expert no domínio;

- **Identificar as Questões de Competência (QC)**: Questões de competência são questões em nível conceitual que a ontologia resultante deve ser capaz de responder. Elas são definidas com base em entrevistas ao *expert* no domínio, usuários ou desenvolvedores. Elas definem a cobertura e a profundidade do escopo de representação da ontologia sobre o domínio modelado. Na Metodologia UPON, há dois tipos de Questões de Competência definidas: (i) orientadas a descoberta de recursos ou conteúdo, e (ii) interoperabilidade semântica entre esquemas distintos. Na metodologia proposta são consideradas apenas questões do primeiro tipo;

[27] De acordo com Guarino et al. (1994), compromissos ontológicos (*ontological commitments*) são mapeamentos feitos entre uma linguagem e algo que pode ser chamado de ontologia.

Figura 5.15 – Visão geral das etapas que compõem o *Context Workflow*.



Fonte: Autoria Própria. Baseada na proposta de De Nicola et al., (2009).

- **Importar Vocabulário Utilizado na Ontologia de Contexto Baseada na Ontologia PiVOn**: Prevê-se a importação do vocabulário da ontologia de contexto nesta etapa. Esta importação é necessária para evitar a redundância de conceitos na fase de modelagem, pois a ontologia PiVOn é utilizada como ontologia de contexto genérica na implementação do *framework* e pelos algoritmos de integração;

- **Criar o Léxico de Referência (LR)**: O léxico de referência é definido a partir do conjunto de conceitos descritos no léxico da aplicação, no vocabulário importado da ontologia PiVOn e nas informações dos documentos utilizados nos processos anteriores. LR é definido como $LR = LA \cap Vocabulário\ da\ Ontologia$;

- **Criar o Glossário de Referência (GR)**: O glossário de referencia é definido a partir da adição de definições informais (frases sobre o conceito) ao LR utilizando linguagem natural;

- **Modelar os Conceitos**: Cada conceito é categorizado através da associação de tipo ao conceito. Os conceitos são categorizados utilizando a ontologia PiVOn como ontologia de topo, ou seja, com os conceitos mais genéricos;

- **Modelar Hierarquias e Relações**: Nesta etapa a categorização de conceitos, descritos em uma taxonomia, é complementada com relações específicas de domínio, relações de agregação e generalização;

- **Selecionar uma Linguagem Formal**: Nesta etapa é selecionada a linguagem para a formalização da ontologia. A etapa é realizada exclusivamente pelo engenheiro de ontologias. Para escolha da linguagem, são considerados o poder de expressividade da linguagem, complexidade computacional para inferência e aceitação da mesma pela comunidade. Recomenda-se a utilização da linguagem OWL-DL nesta metodologia;

- **Formalizar a Ontologia Utilizando a Linguagem Formal**: Nesta etapa os conceitos e relações modelados no *workflow* de projeto (*design workflow*) são formalizados na linguagem OWL-DL. Esta formalização é realizada através da definição de uma ontologia, que importa os conceitos da ontologia PiVOn, formando uma rede de ontologias;

Para realizar a validação da ontologia, um conjunto de verificações devem ser realizadas, de acordo com características distintas: (i) Qualidade sintática e (ii) Qualidade semântica (DE NICOLA et al., 2009).

- **Verificar a Consistência da Ontologia Usando Motor de Inferência**: A verificação da consistência da ontologia deve ser realizada utilizando um motor de inferência. Este motor verifica a existência de contradições na definição da ontologia. Se houver alguma contradição, o motor de inferência informa ao engenheiro de ontologias, que deve revisar o

processo de criação da ontologia. Esta etapa verifica a qualidade sintática e semântica da ontologia;

- **Importar a Ontologia CI**: Além da importação da ontologia PiVOn, faz-se necessária a importação da ontologia CI proposta nesta tese. Assim, os conceitos criados na ontologia de contexto extendido deverão ser subconceitos das classes da ontologia CI ou da ontologia PiVOn;

- **Utilizar o Algoritmo de Integração com a Ontologia CI**: Após a importação dos vocabulários das ontologias citadas anteriormente, os projetistas devem realizar a integração destas ontologias, gerando uma rede de ontologias resultante. Esta rede de ontologias será utilizada nos passos sequentes do *workflow*.

Após a aplicação do *Context Workflow*, é gerada uma rede de ontologias, com consistência verificada através do uso de um motor de inferência e de engenheiros de ontologias através dos processos de verificação da cobertura da ontologia e da resposta de questões de competência. Esta rede de ontologias é utilizada como entrada no *Domain Workflow*, apresentado a seguir.

5.5.2 *Domain Workflow*

O *Domain Workflow* define uma série de processos para a integração de informações de domínio no *framework*. Uma visão geral do *workflow* é apresentada a seguir, na Figura 5.16. O *workflow* é composto pelas seguintes etapas:

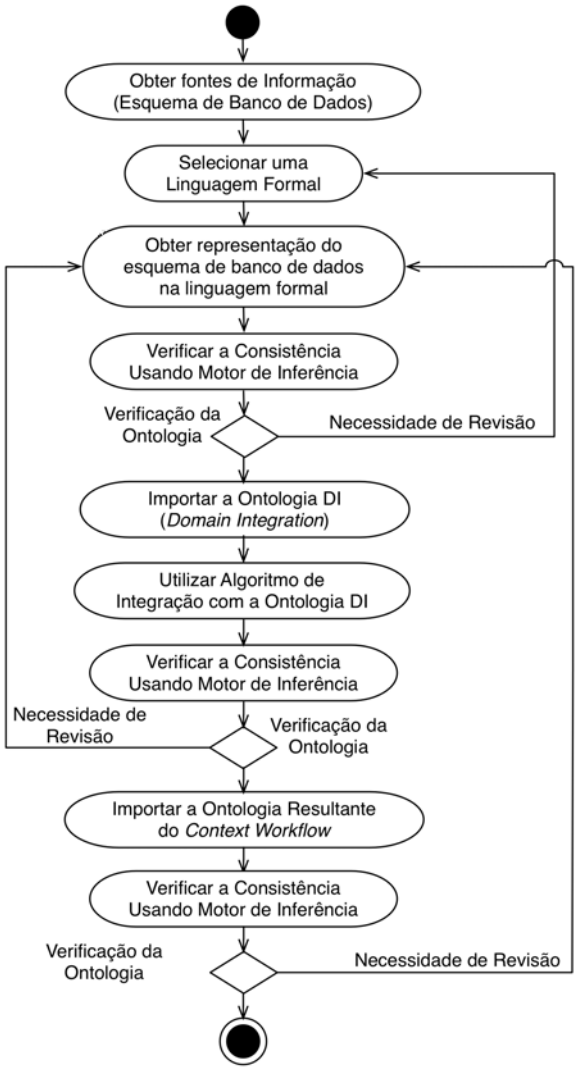
- **Obter fontes de informação**: Nesta etapa o *expert* no domínio deve obter as fontes de informação que serão utilizadas no *framework*. Como o foco desta tese é o *framework* de extensão de consultas em bancos de dados relacionais, deve ser utilizado um esquema de banco de dados como fonte de informação nesta etapa;

- **Selecionar uma linguagem formal**: Nesta etapa, o engenheiro de ontologias realiza a escolha de uma linguagem formal que irá representar o esquema do banco de dados. Como a rede de ontologias gerada no *Context Workflow* é representada na linguagem OWL-DL, é recomendável a escolha das linguagens OWL-DL ou RDF nesta etapa por fins de compatibilidade;

- **Obter representação do esquema de banco de dados na linguagem formal**: Como a representação de contexto no *framework* é feita na linguagem OWL-DL, o esquema de banco de dados utilizado no *framework* precisa ser representado na mesma linguagem. Atualmente, existem ferramentas capazes de realizar a conversão de esquemas relacionais

para arquivos OWL-DL[28], através de mapeamentos R2RML[29]. Foi definido como padrão a utilização da ferramenta *RDBToOnto* (LACLAVIK, 2007);

Figura 5.16 – Visão geral das etapas que compõem o *Domain Workflow*.



Fonte: Autoria Própria.

- **Verificar a consistência utilizando motor de inferência**;

- **Importar e Integrar a Ontologia DI**: Após a primeira verificação, o engenheiro de ontologias realiza a importação da ontologia DI através do algoritmo de importação. A rede de ontologias gerada após esta importação é utilizada no restante do processo;

- **Verificar a consistência utilizando o motor de inferência**;

[28] Uma comparação de ferramentas para a realização de mapeamento entre bancos de dados relacionais e RDF foi realizado por Sahoo et al., (2009).

[29] https://www.w3.org/TR/r2rml/

- **Importar a rede de ontologias resultante do *Context Workflow***: A rede de ontologias gerada no *Context Workflow* é importada e integrada na rede de ontologias gerada no *Domain Workflow*;

- **Verificar a consistência utilizando o motor de inferência**.

Após a realização destas etapas, as definições de contexto e domínio estão interligadas através de uma rede de ontologias. Isto permite que regras de ligacão sejam definidas para interligar as definições no momento da consulta de dados.

5.5.3 *Alignment Workflow*

No *Alignment Workflow* são realizadas as etapas de definição de uma série de contextos que serão levados em consideração no processo de extensão de consultas e das consultas envolvidas nestes processos. Uma visão geral do *workflow* é apresentada a seguir, na Figura 5.17.

O *workflow* é composto pelas seguintes etapas:

- **Obter a ontologia resultante do *Domain Workflow***;

- **Criar lista de contextos relacionados a cada consulta (LCxt)**: Baseado nas questões de competência, o *expert* no domínio de aplicação e o engenheiro de ontologias criam uma lista de: (i) entidades de contexto, (ii) atributos de contexto e (iii) relações semânticas de contexto, que estão relacionadas as questões de competência definidas anteriormente. Estes elementos serão utilizados na definição de regras de ligação;

- **Elencar consultas baseadas nas questões de competência**: Nesta etapa o *expert* no domínio da aplicação elenca, criando uma lista (LC), o conjunto de consultas que são utilizadas no sistema e que podem, de alguma forma, ser contextualizadas;

- **Criar Regras do tipo Domínio E Contexto**: Nesta etapa, caso haja necessidade, são criadas as regras do tipo **Domínio E Contexto**. Estas regras devem ser definidas antes dos outros tipos de regras pois elas criam novos indivíduos na rede de ontologias, permitindo que novas relações semânticas sejam criadas;

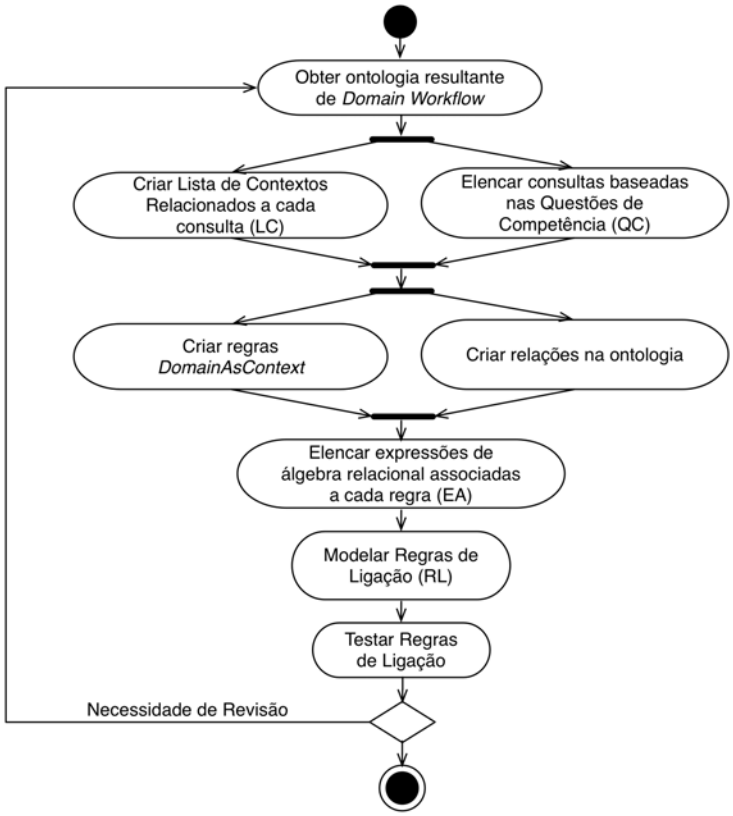
- **Criar Relações na Ontologia**: Caso sejam criadas regras do tipo **Domínio E Contexto**, novos indivíduos são criados na rede de ontologias. Após a criação destes indivíduos, podem ser definidas novas relações semânticas na rede de ontologias com estes novos indivíduos;

- **Elencar expressões de álgebra relacional associadas a cada regra (EA)**: A partir da LCxt e da LC, o *expert* no domínio de aplicação define uma série de expressões em álgebra

relacional (EA) que representam as filtragens que devem ser realizadas de acordo com cada elemento de contexto, para cada consulta;

- **Modelar regras de ligação (RL)**: Utilizando a LCxt, a LC e as EA, o *expert* no domínio modela as regras de ligação utilizando as definições apresentadas anteriormente na seção 5.2;

Figura 5.17 – Visão geral das etapas que compõem o *Alignment Workflow*.



Fonte: Autoria Própria.

- **Testar regras de ligação**: Nesta etapa, as regras modeladas são testadas. Nesta etapa de testes, as regras são testadas apenas em relação a sintaxe e a semântica. Estes testes são feitos através de contextos de interesse, inseridos manualmente, e verificação do retorno das consultas pelo *expert* do domínio. Após a realização da definição das regras de ligação, o conjunto de regras e a rede de ontologias são serializadas.

5.5.4 *Serialization Workflow*

O *Serialization Workflow* trata do conjunto de processos necessários para que as definições realizadas previamente (definição das regras de ligação e da rede de ontologias)

sejam persistidas e possam ser utilizadas pelo algoritmo extensor de consultas. Uma visão gerada do *Serialization Workflow* é apresentada na Figura 5.18.

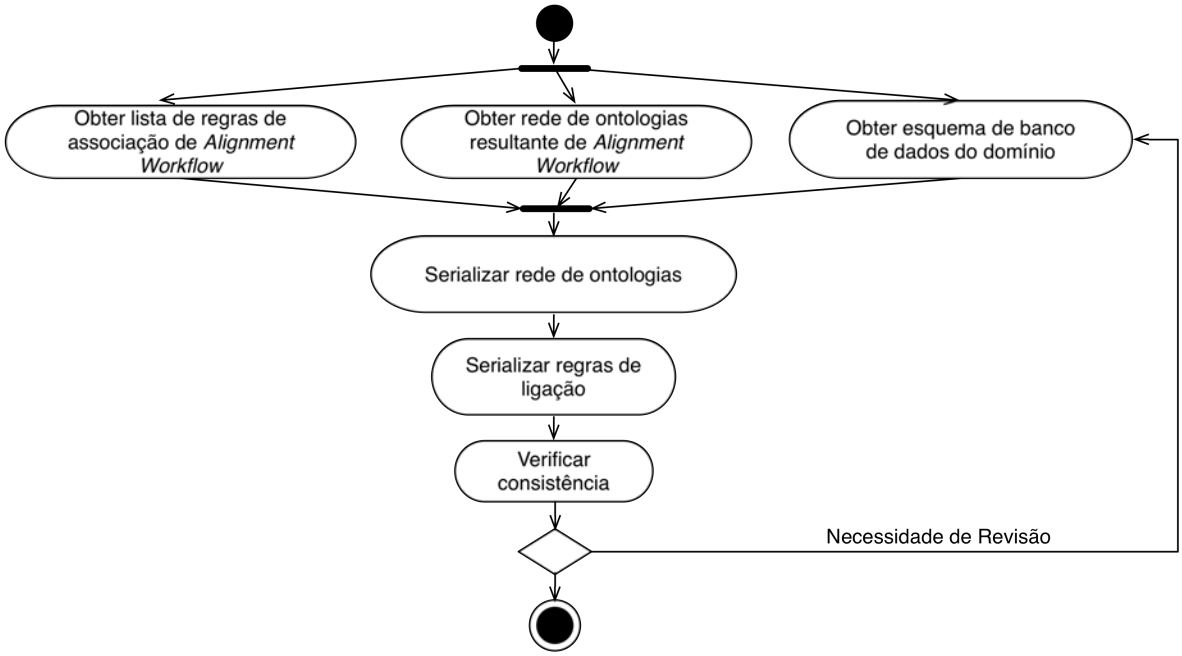
O *workflow* é composto pelas seguintes etapas:

- **Obter as regras de ligacão de *Alignment Workflow***; **Obter rede de ontologias resultante de *Alignment Workflow***; **Obter esquema de banco de dados do domínio**;

- **Serializar rede de ontologias**: Após a etapa de coleta das informações geradas nos processos anteriores, a rede de ontologias é serializada em um formato compatível com o *framework*. Nos protótipos desenvolvidos para suportar o *framework*, foi utilizada a serialização de ontologias no formato JSON-LD;

- **Serializar regras de ligacão**: A serialização de regras de ligacão ocorre na execução dos algoritmos associados a cada regra. Cada um dos algoritmos gera um arquivo no formato JSON-LD que é persistido na mesma instância do banco de dados relacional que armazena as informações de domínio;

Figura 5.18 – Visão geral das etapas que compõem o *Serialization Workflow*.



Fonte: Autoria Própria.

- **Verificar consistência**: Após realizar a persistência da rede de ontologias utilizada pelo *framework*, deve-se verificar a consistência das informações. Para realizar esta verificação, recomenda-se utilizar um motor de inferência.

Após a realização da persistência de informações, a metodologia prevê a execução de testes, para avaliação da aplicação do *framework* no sistema. O *workflow* que descreve a avaliação é apresentado na próxima seção.

5.5.5 *Query Test Workflow*

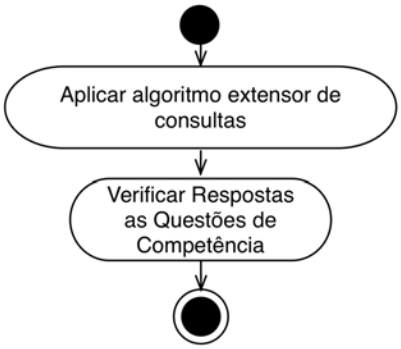
Para realizar a avaliação da aplicação do *framework* no sistema/cenário de aplicação, é necessário integrar o algoritmo extensor de consultas ao sistema e, após, realizar a avaliação da aplicação do *framework* no sistema.

Dois processos devem ser realizados após a aplicação do *framework* no sistema:

- **Aplicar algoritmo extensor de consultas**: A aplicação do algoritmo extensor de consultas é realizado através da integração do módulo integrador no sistema de informação sobre o qual se deseja utilizar o *framework* [30];

- **Verificar respostas as questões de competência**: Nesta etapa, o *expert* no domínio de aplicação verifica, através da realização de consultas ou verificação através dos usuários, se as consultas respondem as questões de competência elencadas na primeira etapa da metodologia. Uma visão geral do *workflow* é apresentada na Figura 5.19.

Figura 5.19 – Visão geral das etapas que compõem o *Query Test Workflow*.



Fonte: Autoria Própria.

5.6 Considerações Finais do Capítulo

Neste capítulo foram apresentados os artefatos que compõem o *framework* proposto nesta tese para realizar a extensão de consultas em bancos de dados relacionais baseadas em

[30] O processo de integração é apresentado no capítulo 6.

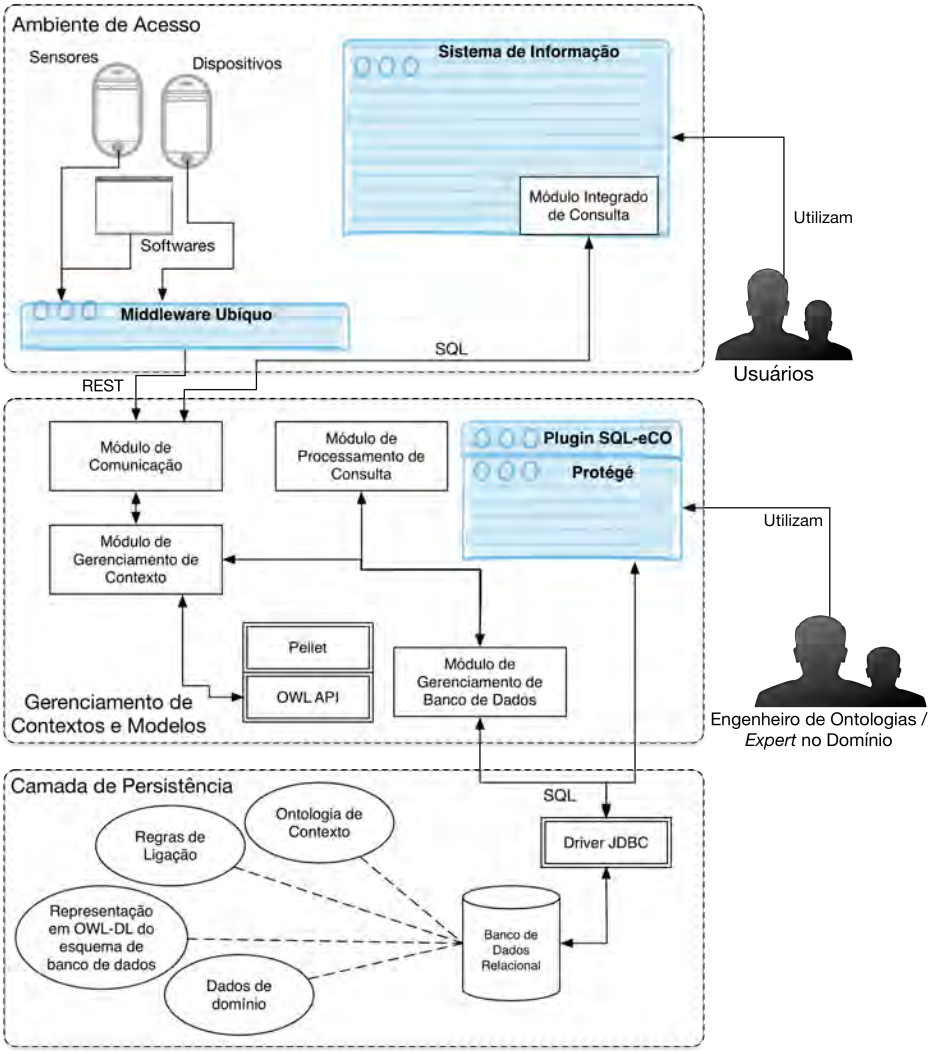
contexto modelado em ontologias. Assim, o *framework* é composto por: (i) Um conjunto de regras de ligação, que possuem algoritmos associados a elas e geram mudanças nas ontologias que representam informações de contexto, (ii) Um algoritmo de extensão de consultas, que a partir de um contexto de interesse informado, realiza a extensão de uma consulta em bancos de dados relacionais, e (iii) Uma metodologia de aplicação do *framework* baseada em uma metodologia de definição de ontologias.

O próximo capítulo apresenta a definição e prototipação de uma arquitetura de software que suporta o *framework* apresentado neste capítulo.

6 ARQUITETURA DE GERENCIAMENTO DO FRAMEWORK

Para realizar a avaliação do *framework* apresentado nesta tese, foi definida uma arquitetura de gerenciamento do *framework* proposto. Desta forma, a integração entre informações de contexto e informações de domínio pode ser utilizada por sistemas já operantes e por novos sistemas. A arquitetura (Figura 6.1) foi definida para atender uma série de requisitos:

Figura 6.1 – Arquitetura proposta para a avaliação do *framework*.



Fonte: Autoria Própria.

(i) Permitir a inferência realizada através de motores de inferência de contextos representados em ontologias no formato OWL-DL;

(ii) Persistir dados de contexto e de domínio em um banco de dados relacional que ofereça suporte a persistência e recuperação de dados no formato JSON;

(iii) Implementar o *framework* de consulta de dados baseada em contexto apresentado nesta tese;

(iv) Oferecer uma interface no padrão REST para a recepção de contextos de interesse de *middlewares* ubíquos.

O principal objetivo da arquitetura é permitir que *middlewares* ubíquos possam informar o contexto do ambiente constantemente à arquitetura, que realiza o armazenamento destas informações e permite a consulta posterior das mesmas, bem como permitir que sistemas de informação realizem consultas de dados através da linguagem SQL e que estas consultas retornem dados de forma contextualizada, utilizando o *framework* apresentado nesta tese.

Esta arquitetura é separada em três níveis: (*i*) ambiente de acesso, (*ii*) camada de gerenciamento de contextos e modelos e (*iii*) camada de persistência. A descrição dos níveis e a apresentação dos protótipos desenvolvidos são apresentados a seguir.

6.1 Ambiente de Acesso

O ambiente de acesso é composto por elementos de software e hardware que geram dados de contexto, que são medidos e inferidos por *middlewares* ubíquos. Além dos *middlewares* ubíquos, sistemas de informação que realizam consultas em bancos de dados relacionais compõem o ambiente físico. No cenário de aplicação desta tese, o sistema de informação utilizado foi um simulador de consultas que simula as consultas realizadas pela plataforma OpenEDX. Este mesmo simulador foi utilizado para representar um *middleware* ubíquo, que envia elementos de contexto de forma constante para a arquitetura de gerenciamento.

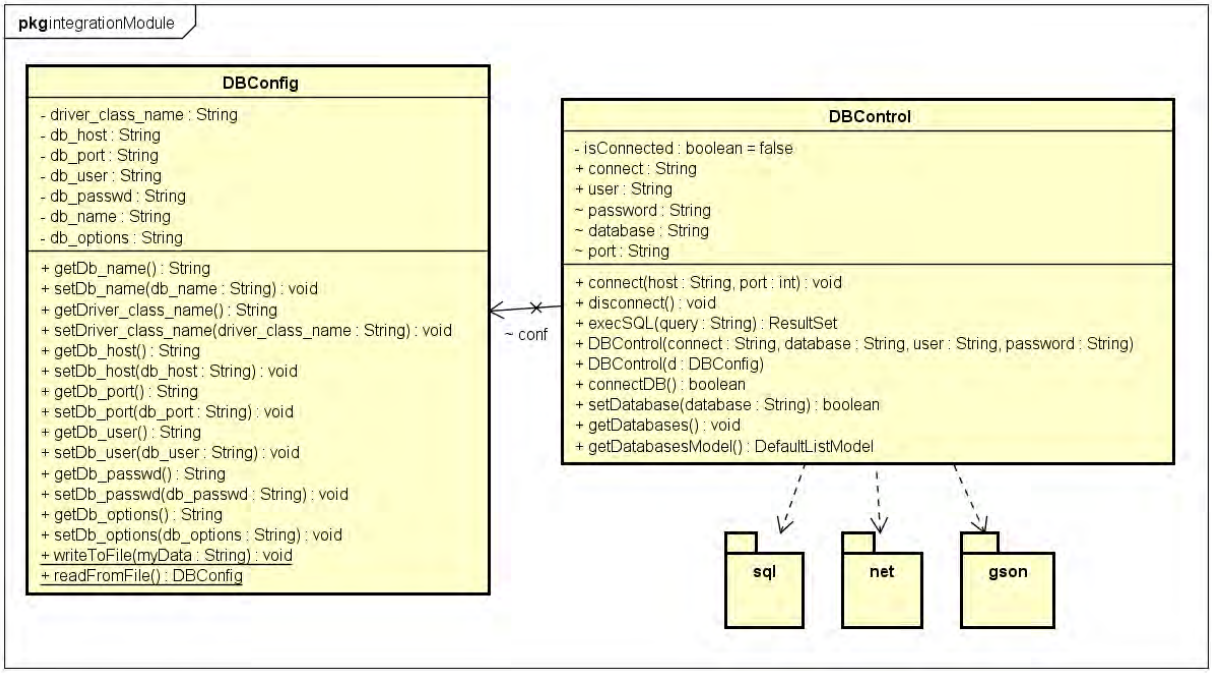
Este ambiente se comunica com os outros níveis da arquitetura através de interfaces padronizadas e amplamente utilizadas. Para a inserção de informações de contexto, usa-se o padrão REST (GRALLA et al., 2003) e, para a consulta de informações baseadas em contexto, utiliza-se o padrão SQL. Este nível da arquitetura *não é o foco desta tese*. Há uma série de implementações de soluções que propõem o gerenciamento e inferência de informações de contexto no nível físico[31]. Como a inferência é realizada pelo *middleware*

[31] Exemplos de implementações: *OpenHAB*: http://www.openhab.org/ e a proposta apresentada por Kazzak & Rychly (2013).

ubíquo, a arquitetura apresentada neste trabalho apenas armazena contextos de interesse, sem a necessidade de inferir novamente sobre estes contextos.

Um módulo, definido na linguagem Java, deve ser importado no sistema de informação e executa os mesmos métodos definidos no *driver* JDBC para consulta de dados. O módulo desenvolvido, para ser integrado no sistema de informação, possui uma classe principal *DBControl* (Figura 6.2). Esta classe contém os métodos necessários para a conexão e a realização de consultas contextualizadas gerenciadas pela arquitetura. No momento em que o método *connect()* é executado, o módulo estabelece uma conexão via *socket* com o módulo de comunicação da arquitetura. A partir deste momento, o método *execSQL(String sql_statement)* pode ser utilizado. Este método recebe uma expressão na linguagem SQL, representada em uma *string*, envia a expressão para o módulo de comunicação e retorna um objeto da classe *ResultSet* (assim como no *driver* JDBC), que é enviado pelo módulo de comunicação.

Figura 6.2 – Diagrama de classe do módulo integrado ao sistema de informação.



Fonte: Autoria Própria.

A classe *DBConfig* gerencia as configurações do módulo em relação aos *hosts*, portas de comunicação e configurações de comunicação com o banco de dados. O módulo foi compilado em um arquivo do tipo *jar*, que é importado no sistema de informação. O ambiente físico envia e espera respostas da arquitetura através da comunicação com o nível de Gerenciamento de Contexto e Modelos, apresentado a seguir.

6.2 Gerenciamento de Contextos e Modelos

Este nível da arquitetura oferece uma série de módulos que gerenciam o *framework* apresentado na tese e oferece interfaces de comunicação que podem ser utilizadas por outros programas e recursos de outras camadas da arquitetura. Assim, este nível atua como um intermediário entre a comunicação dos sistemas de informação e o *driver* JDBC que realiza as operações em bancos de dados relacionais. Os módulos desta camada da arquitetura foram separados em pacotes, de acordo com a funcionalidade das classes. Muitas das funcionalidades presentes nos módulos que compõem esta camada foram implementados no *plugin* integrado à ferramenta Protégé. Assim, o *plugin* funciona como uma ferramenta de auxílio *standalone* em relação à arquitetura. O *plugin* é utilizado para auxiliar no processo de execução da metodologia, enquanto a arquitetura em si é o protótipo de implementação do *framework.*

6.2.1 Plugin SQL-eCO

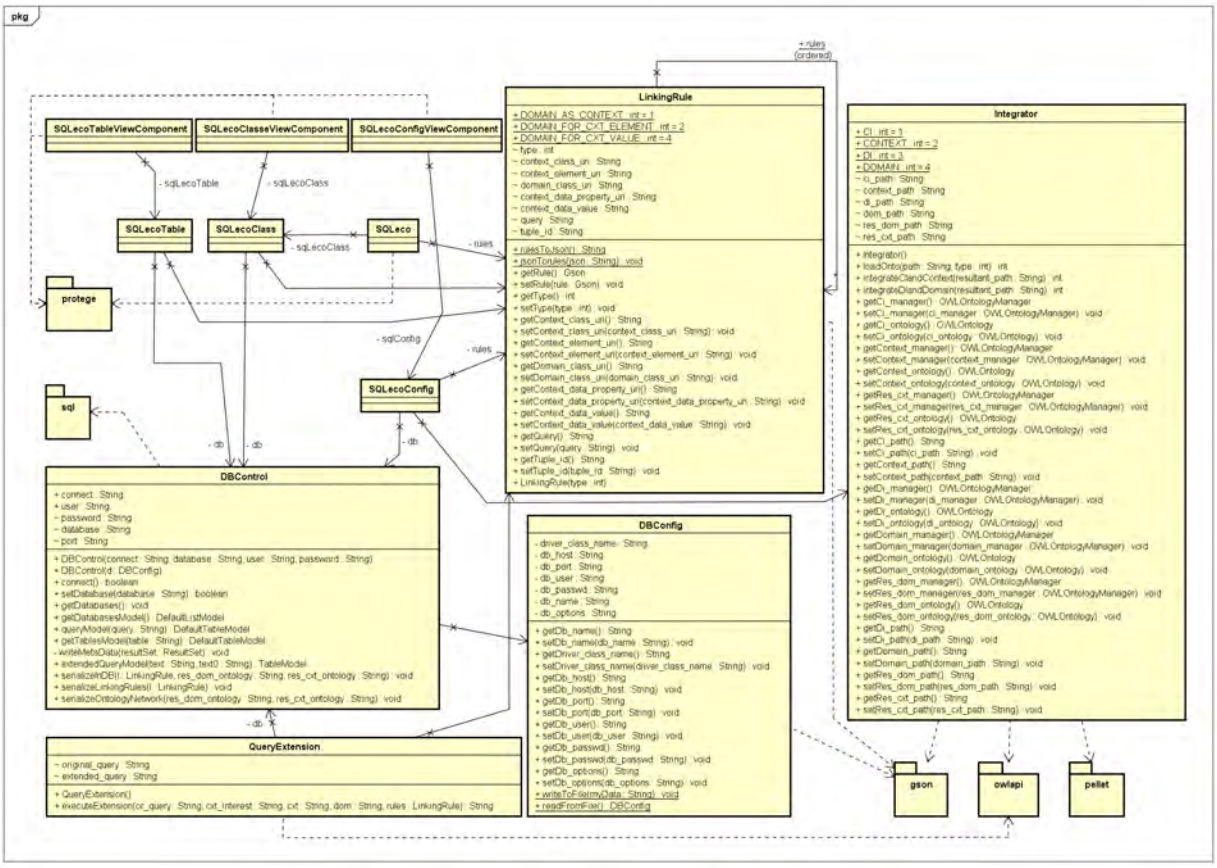
O *plugin* SQL-eCO (*SQL extended by Context-Awareness based on Ontologies*) foi desenvolvido para oferecer uma interface de auxílio a engenheiros de ontologias e *experts* no domínio de aplicação para a criação e gerenciamento de regras de ligação propostas pelo *framework* apresentado nesta tese. Através do *plugin* é possível utilizar as implementações dos algoritmos associados às regras e de integração sem alterar o esquema de banco de dados, na aplicação da metodologia proposta na tese. Após a verificação dos modelos, através da verificação utilizando o motor de inferência Pellet, é possível exportar todas as definições para o banco de dados de domínio, onde as definições podem ser utilizadas pela arquitetura prototipada. O *plugin* foi desenvolvido na linguagem Java, utilizando as APIs disponíveis pelo *software* Protégé. A Figura 6.3 apresenta o diagrama de classes do *plugin* SQL-eCO[32].

As classes sem atributos e métodos apresentadas no digrama realizam o controle da interface com o usuário. As classes *DBConfig* e *DBControl* oferecem métodos para o controle das configurações e operações no banco de dados relacional. As classes *QueryExtension*, *Integration* e *LinkingRule* implementam os algoritmos apresentados no Capítulo 5 e que compõem o *framework* apresentado nesta tese. As classes do *plugin* utilizam métodos

[32] Os atributos e métodos relacionados ao controle da interface com o usuário foram ocultados para facilitar a visualização.

definidos nas APIs Gson[33], OWL API[34] e Pellet *Reasoner*[35]. Estas APIs foram integradas ao projeto do *plugin*, no *software* Protégé.

Figura 6.3 – Diagrama de classes do *plugin* SQLeco.



Fonte: Autoria Própria.

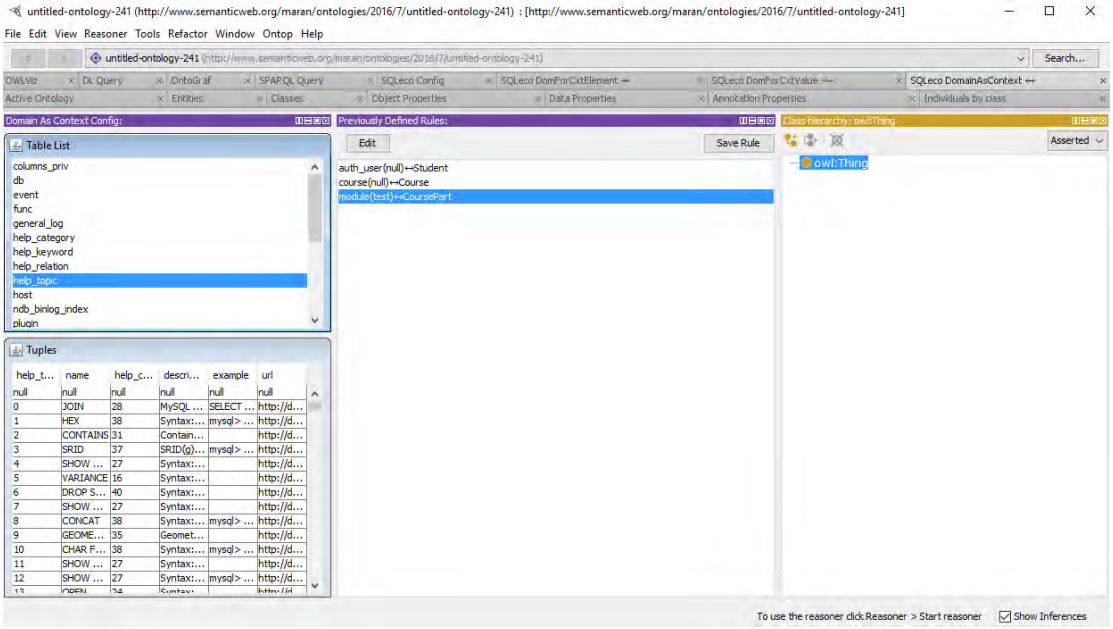
A interface do *plugin* é composta por quatro abas, que são utilizadas para modelar os diferentes tipos de regras de ligação (três abas), configurar a conexão com o banco de dados relacional e realizar testes. Na aba *SQL-eCO DomainAsContext* (Figura 6.4) é possível modelar regras do tipo *Domain As Context*, assinalando a classe relacionada à regra (na lista do lado direito), a tabela relacionada à regra (na lista superior do lado esquerdo) e uma ou mais tuplas correspondentes à regra na lista inferior no lado esquerdo. Após a marcação das informações, o usuário salva as informações relacionadas à regra, que aparecem na lista de regras (no centro da interface).

[33] https://github.com/google/gson

[34] http://owlapi.sourceforge.net

[35] https://github.com/Complexible/pellet

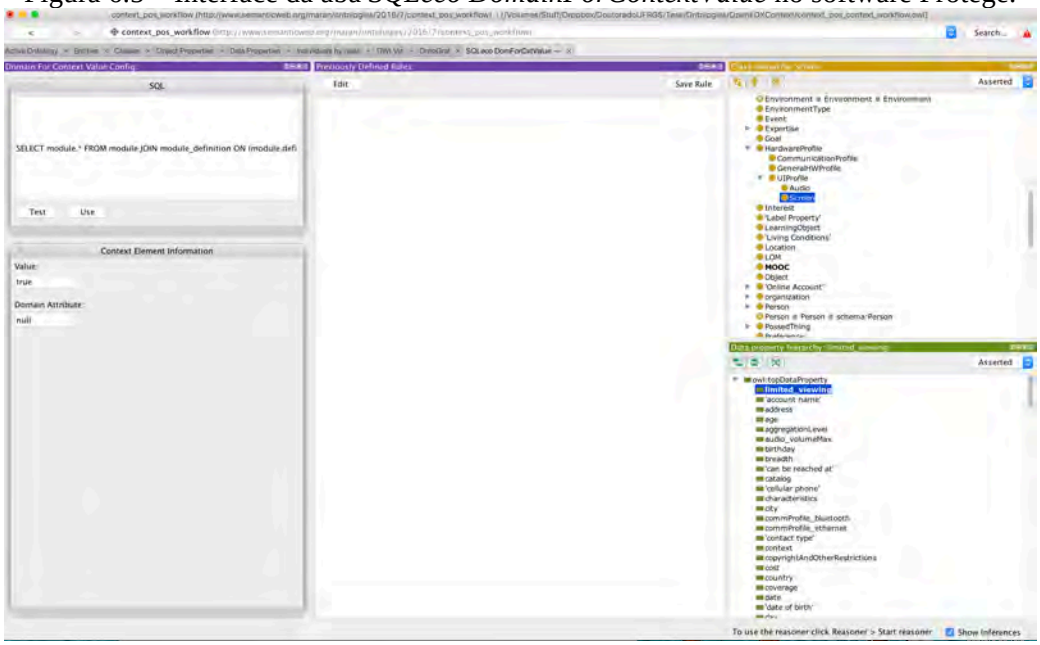
Figura 6.4 – Interface da aba *SQLeco DomainAsContext* no software Protégé.



Fonte: Autoria Própria.

Na aba *SQL-eCO DomainForContextValue* (Figura 6.5) é possível modelar regras do tipo *Domain For Context Value*, assinalando a classe relacionada à regra (na lista do lado direito), a tabela relacionada à regra (na lista superior do lado esquerdo) e uma ou mais tuplas correspondentes à regra na lista inferior no lado esquerdo. Após a marcação das informações, o usuário salva as informações relacionadas à regra, que aparecem na lista de regras (no centro da interface).

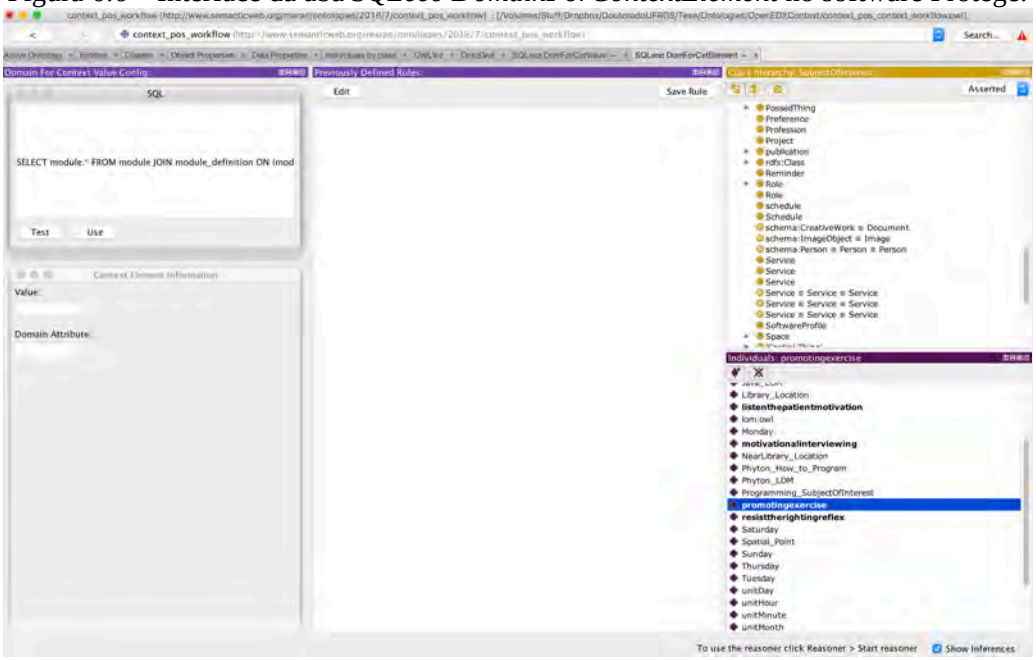
Figura 6.5 – Interface da aba *SQLeco DomainForContextValue* no software Protégé.



Fonte: Autoria Própria.

Na aba *SQL-eCO DomainForContextElement* (Figura 6.6) é possível modelar regras do tipo *Domain For Context Element*, assinalando a classe relacionada à regra (na lista do lado direito na parte de cima) e o indivíduo de contexto relacionado à regra (na lista do lado direito, na parte de baixo). Após a marcação destas informações, o usuário escreve a consulta relacionada a esta regra, podendo testar o resultado da consulta com o botão *Test*. Após, ele salva as informações relacionadas à regra, que aparecem na lista de regras (no centro da interface).

Figura 6.6 – Interface da aba *SQLeco DomainForContextElement* no software Protégé.



Fonte: Autoria Própria.

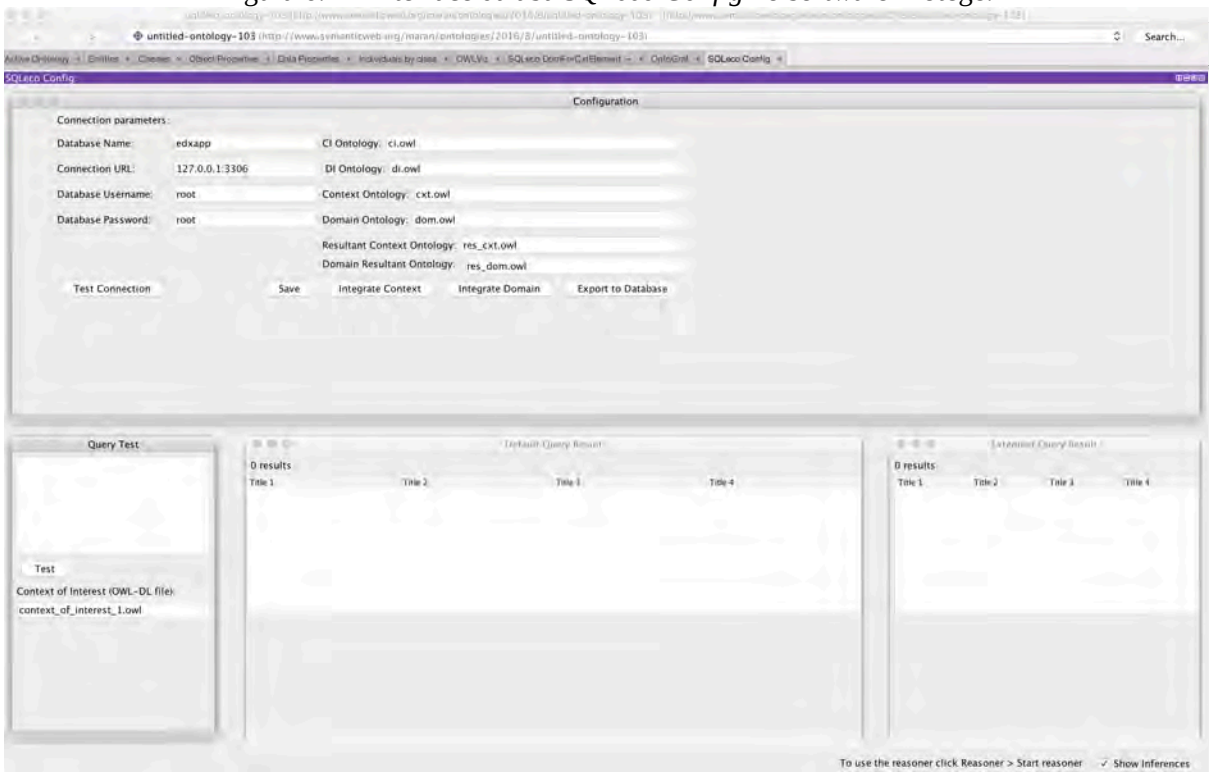
Na aba *SQL-eCO Config* (Figura 6.7) é possível configurar a conexão com o banco de dados utilizado para a realização de testes com as regras de ligação. Além disso, esta interface oferece acesso aos algoritmos apresentados no capítulo 5 para a integração de ontologias e realização de testes de consultas, onde é informado o contexto de interesse, modelado em um arquivo OWL-DL, e uma consulta relacional no formato SQL.

Após a definição de regras e testes de consultas feitas no *plugin*, o usuário pode exportar as definições de regras e a rede de ontologias para o banco de dados relacional. Essa exportação é feita em duas etapas distintas: (*i*) são criadas as tabelas no esquema para armazenamento das definições[36] e as regras de ligação, definidas em JSON são persistidas no

[36] O esquema de tabelas utilizado pelo modelo foi apresentado no capitulo 5.

banco de dados e (*ii*) A rede de ontologias é serializada no formato JSON utilizando a API OWLAPI-JSONLD[37] e armazenada no banco de dados relacional. A forma de gerenciamento da comunicação é apresentada na próxima seção.

Figura 6.7 – Interface da aba *SQLeco Config* no software Protégé.



Fonte: Autoria Própria.

6.2.2 Comunicação

O módulo de comunicação oferece interfaces de comunicação para sistemas de informação externos e *middlewares* ubíquos. O módulo é composto por duas interfaces de comunicação: (i) uma interface baseada em um serviço REST[38], que permite com que *middlewares* ubíquos realizem operações de POST em um recurso, enviando assim contextos de interesse (via um arquivo XML que descreve os indivíduos que compõem o contexto de interesse) e (ii) um *socket*[39] de comunicação que permite que o módulo integrado ao sistema de informação, desenvolvido na linguagem Java, realize a comunicação com a arquitetura.

[37] https://github.com/stain/owlapi-jsonld

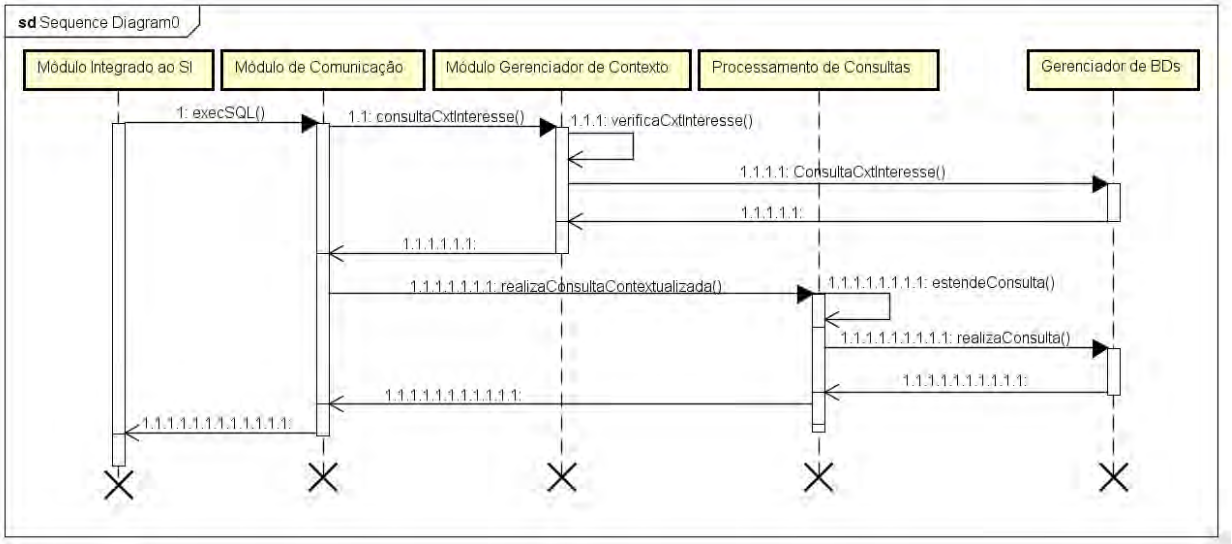
[38] O serviço foi implementado no servidor *Grizzly*: https://grizzly.java.net/

[39] https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html

O *socket* implementado no módulo de comunicação da arquitetura espera mensagens com uma requisição de operação do tipo *SELECT*. O diagrama de sequência apresentado na Figura 6.8 apresenta a sequência de eventos ocorridos na arquitetura a cada requisição de consulta. Inicialmente, o módulo integrado no sistema de informação envia uma requisição de execução de uma expressão da consulta, no formato SQL, para o módulo de comunicação da arquitetura via *socket*.

O módulo de comunicação realiza uma requisição ao módulo gerenciador de contexto de qual contexto de interesse está associado a consulta. A associação é verificada pelo *id* do usuário e pelo horário inicial e final das situações que envolvem este usuário e o horário da consulta. O módulo gerenciador de contextos verifica se há um contexto de interesse que pode ser associado à consulta, realizando uma consulta no banco de dados. Se há um contexto de interesse que pode ser associado a consulta, ele é informado ao módulo de comunicação, que o repassa juntamente com a consulta para o módulo de processamento de consultas. Este módulo por sua vez realiza a extensão da consulta em SQL, retornando um conjunto de valores que representam as tuplas resultantes da consulta ao módulo de comunicação, que repassa as informações ao módulo integrado ao SI via *socket*.

Figura 6.8 – Diagrama de sequência das operações realizadas a cada nova consulta contextualizada.



Fonte: Autoria Própria.

Além da interface com o módulo integrado ao SI, o módulo de comunicação oferece uma interface REST para que *middlewares* ubíquos enviem elementos de contexto para a arquitetura de persistência e recuperação. O envio é feito através de uma requisição *POST* no

endereço `http://localhost:8080/contextarch/cxt`. Esta requisição contém o arquivo XML que descreve as entidades do contexto de interesse na ontologia de contexto, representadas em OWL-DL. Um exemplo de contexto de interesse descrito em um arquivo XML é apresentado na Figura 6.9.

Figura 6.9 – Representação em XML de um contexto de interesse.

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
xml:base="http://www.semanticweb.org/maran/ontologies/2016/7/context_pos_workflow"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
ontologyIRI="http://www.semanticweb.org/maran/ontologies/2016/7/context_pos_workflow">
    <Prefix name=""
IRI="http://www.semanticweb.org/maran/ontologies/2016/7/context_pos_workflow#"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Import>http://www.semanticweb.org/maran/ontologies/2016/7/context_</Import>
    <Declaration>
        <NamedIndividual IRI="#John"/>
    </Declaration>
    <Declaration>
        <NamedIndividual IRI="#in_class_break"/>
    </Declaration>
    <ClassAssertion>
        <Class
IRI="http://www.semanticweb.org/maran/ontologies/2014/11/pivon_user#User"/>
        <NamedIndividual IRI="#John"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class
IRI="http://www.ccsul.ufsm.br/~maran/ontologies/pivon_user.owl#UserSituation"/>
        <NamedIndividual IRI="#in_class_break"/>
    </ClassAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty
IRI="http://www.ccsul.ufsm.br/~maran/ontologies/pivon_user.owl#user_isIn_userSit"/>
        <NamedIndividual IRI="#John"/>
        <NamedIndividual IRI="#in_class_break"/>
    </ObjectPropertyAssertion>
</Ontology>
<!-- Generated by the OWL API (version 4.2.5.20160517-0735)
https://github.com/owlcs/owlapi -->
```

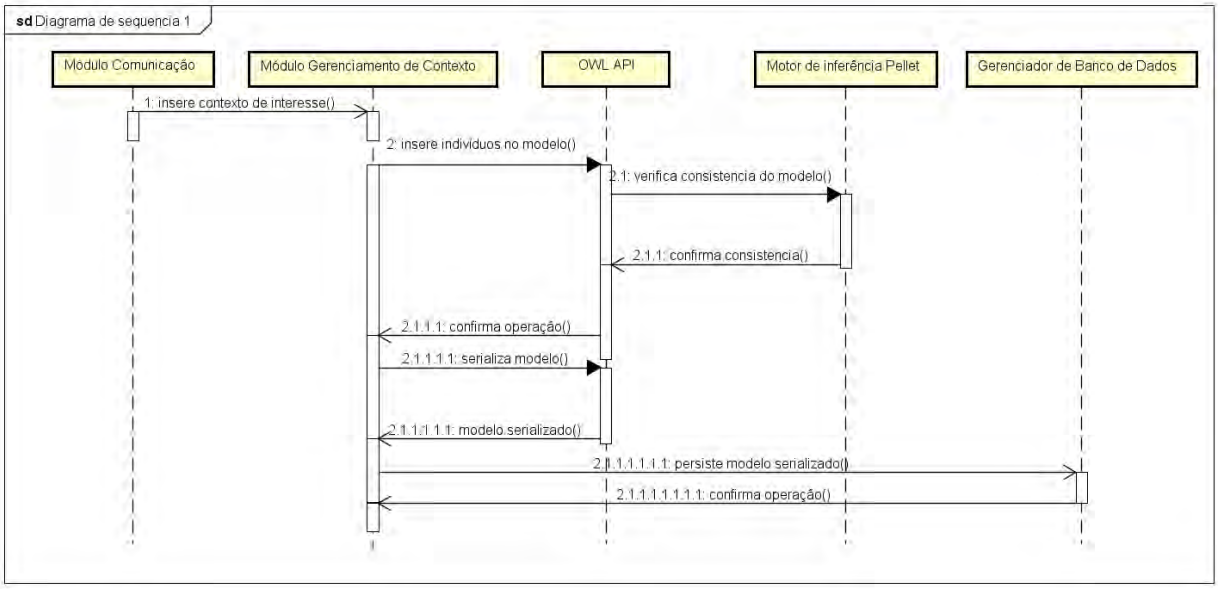
Fonte: Autoria Própria.

Como pode ser observado na Figura 6.9, o arquivo XML que contém o contexto de interesse representa apenas os indivíduos e suas relações que, juntos, compõem o contexto de interesse. As definições de classes são declaradas nas ontologias informadas no cabeçalho do

arquivo. O contexto de interesse informa que um usuário chamado *John* está atualmente em uma situação *in_class_break*.

O digrama de sequência apresentado na Figura 6.10 mostra a sequência de eventos gerados na arquitetura a cada nova recepção de contextos de interesse. Inicialmente, o módulo de comunicação envia ao módulo de gerenciamento de contexto o arquivo em XML com a descrição do contexto de interesse através do método *insertCxtofInterest()*.

Figura 6.10 – Diagrama de sequência na operação de inserção de um novo contexto de interesse.



Fonte: Autoria Própria.

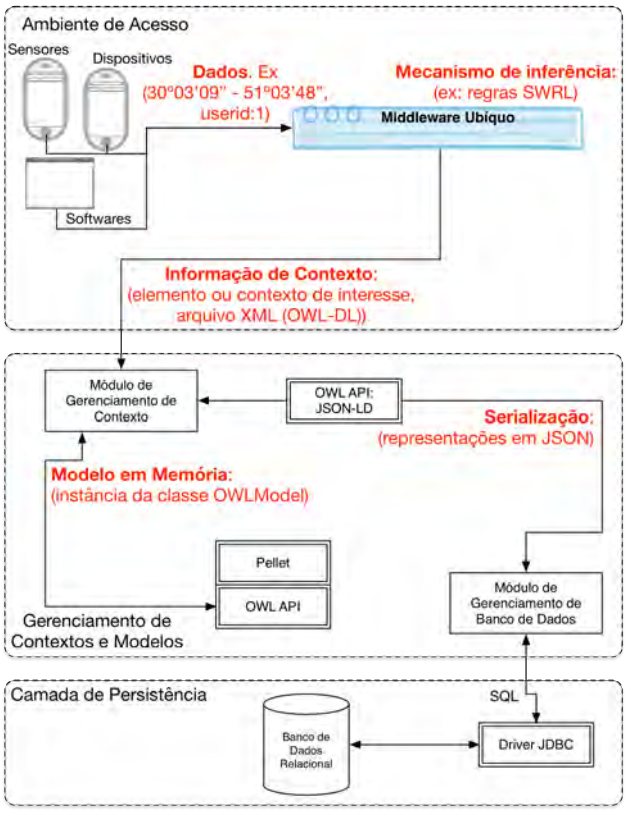
O módulo gerenciador de contexto insere somente os indivíduos e propriedades que não estavam previamente definidos na ontologia. Esta inserção é realizada no modelo que representa a ontologia em memória, gerenciado pela API OWL-API. Após a inserção dos novos indivíduos na ontologia, a consistência da mesma é verificada através do motor de inferência Pellet.

Se a consistência da ontologia é validada, o módulo de gerenciamento de contexto realiza a serialização do modelo no formato JSON através da API OWL-API JSON-LD e insere as definições serializadas no banco de dados relacional através do módulo gerenciador de banco de dados.

6.2.3 Gerenciamento de Contexto

O módulo de gerenciamento de contexto realiza a verificação da consistência dos elementos da rede de ontologias de contexto e oferece serviços relacionados ao gerenciamento de contexto aos outros módulos, como por exemplo, a verificação de consistência do modelo que representa a ontologia de contexto em memória e a consulta de contextos de interesse relacionados a uma consulta relacional. O diagrama da Figura 6.11 apresenta um esquema de funcionamento do módulo de gerenciamento de contexto da arquitetura.

Figura 6.11 – Utilização de informações de contexto na arquitetura.



Fonte: Autoria Própria.

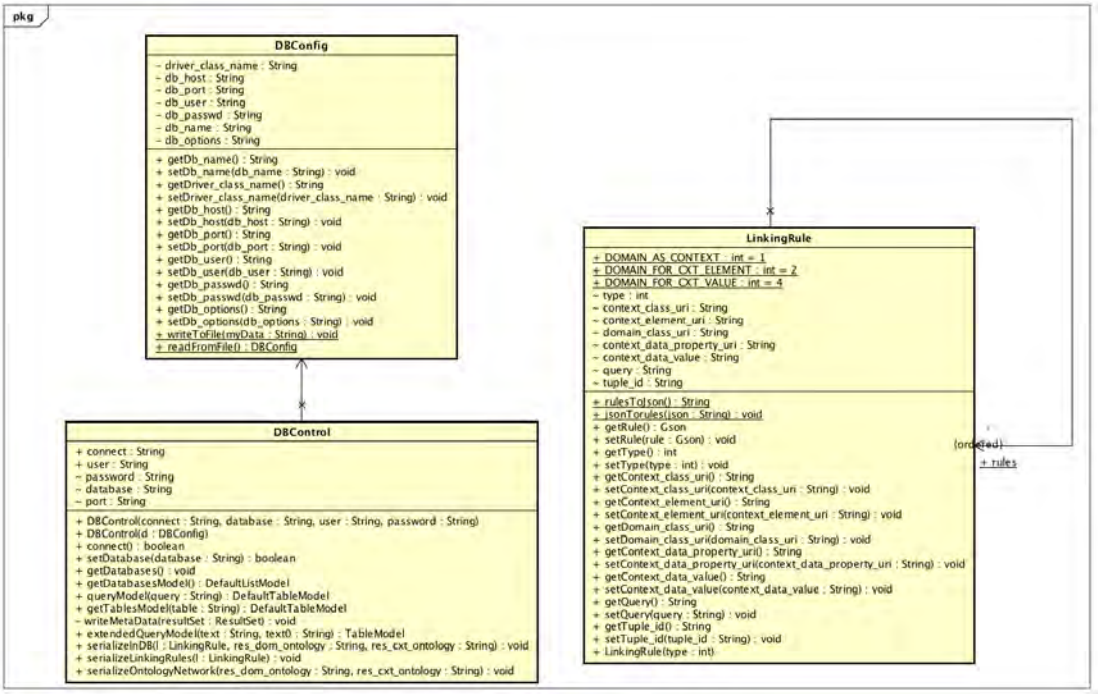
O módulo gerencia a rede de ontologias utilizada no modelo em memória, através da API OWL-API. A serialização da rede de ontologias para o formato JSON também é feita pela mesma API. A consistência do modelo em memória é realizada pela API do motor de inferência Pellet[40]. O módulo de gerenciamentoo de contexto também realiza o controle das regras de ligação utilizadas pelo *framework*. Cada uma das regras de ligação é representada no módulo por um objeto da classe *LinkingRule*.

O modelo que representa a rede de ontologias utilizada pelo *framework* apresentado nesta tese é gerenciado em memória e representada como um objeto da classe *OWLModel*. A cada novo recebimento de um elemento de contexto ou contexto de interesse, representado em OWL-DL no formato XML, uma nova instância da classe *OWLModel* é criada, as representações do arquivo são importadas para esta instância e sua consistência é verificada. Se há consistência nestas definições, é realizado um *merge*[41] entre esta instância e a instância que representa a rede de ontologias já utilizada na arquitetura. Após a realização do *merge* entre os modelos, o módulo realiza a serialização das definições no formato JSON e o resultado da serialização é persistido no banco de dados.

6.2.4 Gerenciamento de Banco de Dados

O módulo de gerenciamento de banco de dados realiza as operações no banco de dados relacional utilizado na arquitetura através do *driver* JDBC. O módulo é composto por três classes que são utilizadas pelos outros módulos que compõem a arquitetura. O diagrama das classes do módulo é apresentado a seguir, na Figura 6.12.

Figura 6.12 – Diagrama de classes do módulo de Gerenciamento de banco de dados.



Fonte: Autoria Própria.

[40] https://github.com/stardog-union/pellet

[41] O método de *merge* entre instâncias de *OWLModel* é definido na API OWL-API.

A classe *DBConfig* possui métodos para gerenciar as configurações de conexão com o banco de dados em relação ao armazenamento e recuperação das definições de conexão, enquanto a classe *DBControl* possui métodos para realizar a conexão e consulta no banco de dados relacional utilizando o *driver* JDBC. A classe *LinkingRule* representa uma regra de ligação. Cada objeto desta classe é serializado para uma representação em JSON persistida no banco de dados relacional.

6.2.5 Processamento de Consultas

O módulo de processamento de consultas implementa o algoritmo de extensão de consultas apresentado anteriormente no capitulo 5. A extensão de consultas é realizada em todas as requisições do tipo *SELECT*. Para selecionar o contexto de interesse associado à consulta, o módulo de processamento de consultas realiza uma consulta ao módulo de gerenciamento de contexto para saber o contexto de interesse referente à consulta. O módulo de gerenciamento de contexto informa ao módulo processador de consultas o contexto de interesse da consulta.

O contexto de interesse de uma consulta relacional é definido por $CxtInt = \{< ind, obj\_prop, data\_prop >\}$, ou seja, um conjunto de indivíduos da ontologia e suas relações semânticas com outros indivíduos e suas propriedades de dado. Um destes indivíduos pertence à classe *Pivon:UserSituation*, que define a situação de um usuário. Para ser utilizado na extensão de consulta, são utilizados os indivíduos e relações semânticas relacionadas ao mesmo usuário que realiza a consulta, onde $tis < tc \wedge (tfs = 0 \vee tfs > tc)$, sendo *tis* o *timestamp* inicial da situação, *tc* o *timestamp* da consulta relacional e *tfs* o *timestamp* final da situação, onde o valor de *tfs* é 0 quando a situação não chegou ao seu final ainda.

Quando um pedido de consulta é recebido pela arquitetura, o módulo de processamento de consultas consulta o módulo de gerenciamento de contexto por contextos de interesse relacionados ao mesmo usuário envolvido na consulta. Os contextos de interesse relacionados devem ter tempos iniciais e finais que obedecem às regras definidas anteriormente. O módulo de gerenciamento de contexto informa ao módulo de processamento de consultas os contextos de interesse relacionados à consulta.

Assim, o módulo de processamento de consultas executa o algoritmo extensor de consultas definido nesta tese, verificando se algum dos elementos de contexto que compõem o contexto de interesse possui alguma regra de ligação associada. Caso algum elemento de

contexto possua alguma regra de ligação associada, o módulo de processamento de consultas estende a consulta relacional com a expressão de consulta relacionada à regra de ligação.

6.3 Gerenciamento de Persistência

O gerenciamento de persistência dos dados é realizado pelo SGBD utilizado pela arquitetura. Como as definições utilizadas na rede de ontologias são serializadas no formato JSON e são persistidas no mesmo esquema de banco de dados utilizado no domínio, a comunicação com o banco de dados é realizado através da utilização de um *driver* JDBC.

Há uma série de serviços e funcionalidades que podem ser estendidas das definições da arquitetura apresentadas nesta tese. Foram implementados os protótipos necessários para a avaliação do *framework* [42]. Após a implementação dos protótipos de serviços que compõem a arquitetura, foi definido um cenário de aplicação do *framework* e o mesmo foi aplicado utilizando a metodologia e os protótipos desenvolvidos.

[42] Propostas de extensões para a arquitetura são apresentadas na seção de trabalhos futuros (capitulo 8).

7 AVALIAÇÃO E DISCUSSÃO

A avaliação do *framework* foi realizada através da aplicação do mesmo em um cenário de aplicação baseado no cenário motivador. O cenário de aplicação utilizado na avaliação é apresentado a seguir.

7.1 Definição do Cenário de Aplicação

A educação vem evoluindo constantemente em relação aos processos de ensino. Recentemente, tecnologias como a computação móvel e a internet tem contribuído para uma maior intensidade de disseminação de informação e materiais de suporte à educação (COOPER; SAHAMI, 2013). Como há um grande número de informações e materiais disponíveis para professores e alunos, faz-se necessária a existência de filtros nas consultas realizadas sobre estas informações, desta forma restringindo-as ao domínio de estudo do aluno e ao contexto informado no momento da consulta (COOPER; SAHAMI, 2013).

Atualmente, professores montam e distribuem materiais em Ambientes Virtuais de Aprendizagem (AVAs). Estes materiais são distribuídos e atividades são executadas nestes ambientes por estudantes sempre da mesma forma, desconsiderando a importância das variáveis de contexto envolvidas na realização destas tarefas.

Este problema é acentuado em MOOCs, principalmente porque existem variações maiores em relação a questões culturais e geográficas se comparadas a um curso a distância tradicional. Esta variedade de fatores influencia diretamente na baixa taxa de conclusão nestes cursos, que tem sido de 5 a 9% (DILLENBOURG et al., 2014). Desta forma, é necessário que informações de contexto sejam levadas em consideração na forma como estes materiais são distribuídos aos alunos destes cursos (COOPER; SAHAMI, 2013) (GUTIÍRREZ-ROJAS et al., 2014). O seguinte cenário foi considerado para aplicar a abordagem definida nesta tese. Ela é baseada no cenário motivador apresentado no capítulo 2:

Vamos imaginar que uma universidade utiliza um middleware ubíquo que gerencia os recursos educacionais presentes nesta universidade. Este middlware utiliza uma ontologia como modelo de representação de contexto. Esta ontologia contém um conjunto de axiomas que, associados a um conjunto de processos gerenciados pelo middleware, permitem que dados coletados do ambiente sejam agregados, e após a realização de inferências sejam utilizados como informação de contexto. Assim, o gerenciamento de contexto de interesse

neste middleware ubíquo é limitado a inferir as situações nas quais alunos e professores se encontram. Esta universidade possui ainda um sistema de recomendação que apresenta avisos aos estudantes sobre recursos educacionais que possam ser do seu interesse. Portanto, o contexto de interesse de cada estudante e professor é inferido e repassado ao sistema de recomendação, que recomenda ao aluno ou professor um recurso de interesse disponível na universidade. O middleware ubíquo utiliza a ontologia PiVOn para descrever elementos de contexto e contextos de interesse.

Um MOOC preparado por uma equipe de professores da área de psicologia e medicina de uma universidade é composto de casos de entrevista motivacional (EM). Entrevista Motivacional é uma abordagem centrada no paciente para demonstrar mudanças comportamentais para aumentar a efetividade no tratamento de doenças, como obesidade, tabagismo e depressão. O grupo de técnicas que compõem uma EM tem sido constantemente estudadas em uma grande variação de comportamentos relacionados à saúde (COLE et al., 2011). *Para preparar o MOOC sobre entrevista motivacional, os professores montam o curso com uma estrutura composta de tópicos, onde cada tópico consiste em um material de apoio (vídeo, documento ou apresentação) e um conjunto de perguntas. Se o aluno responde corretamente as perguntas relacionadas a aquele tópico, ele pode ir para o próximo, onde é apresentado a novas situações de entrevista motivacional. O aluno conclui o curso quando completa todas as tarefas relacionadas ao aprendizado básico de conceitos e 50% dos exercícios relacionados a casos de entrevista motivacional específicos, relacionados a técnicas específicas de entrevista motivacional aplicadas a diferentes contextos (doença a ser tratada, forma de motivação, entre outros). Este MOOC é implementado na plataforma OpenEDX.*

João é um estudante do curso de saúde coletiva na universidade que possui o middleware ubíquo e o sistema de recomendação de recursos. No momento da matrícula, João registrou seu smartphone no sistema da universidade e a partir daí passou a receber recomendações de recursos. João é brasileiro e mora na cidade de Porto Alegre.

Atualmente, João cursa o 6º semestre, e em uma aula da disciplina Psicologia Aplicada à Saúde, o sistema de recomendação utilizado pela universidade emite um alerta a João no seu smatphone recomendando a inscrição no MOOC sobre Entrevistas Motivacionais. Durante o intervalo da aula, João visualiza o alerta e se inscreve no MOOC de Entrevista Motivacional. Neste momento, o MOOC pede a João autorização para receber informações de contexto gerenciadas pelo middleware ubíquo da universidade. João autoriza e o MOOC passa a receber atualizações constantes de contextos de interesse.

Durante o intervalo, João acessa o MOOC através de seu smartphone e observa a existência de 3 tópicos de introdução ao conceito de Entrevista Motivacional. O curso possui mais tópicos de introdução, porém, o MOOC apresenta somente tópicos que não possuem vídeos associados, ou que possuem vídeos com duração menor que 10 minutos, pois o tempo de intervalo não permitiria que João olhasse um vídeo de caso de entrevista motivacional maior que este tempo de duração.

Após o intervalo, João volta para a aula e mostra o MOOC para alguns colegas, que também se inscrevem no curso. Como João está na aula, o MOOC apresenta apenas uma introdução ao curso e informações gerais sobre quais casos João já completou os questionários.

No laboratório de informática, utilizando um desktop, João acessa o portal do aluno e é direcionado a um questionário sobre suas áreas de interesse. João assinala que possui interesse nas áreas de promoção da saúde, educação física e sistemas de saúde. Estas áreas de interesse foram registradas por João em um questionário aplicado durante o curso e estão associadas ao perfil de João. João acessa o MOOC novamente. Neste acesso, João é apresentado aos módulos do curso que apresentam os conceitos básicos de entrevista motivacional. João assiste aos vídeos associados aos módulos e responde os questionários. Após a realização destas etapas, João é apresentado a uma nova série de cenários de entrevista motivacional.

7.2 Dados Utilizados nos Experimentos

Para realizar os experimentos de aplicação do *framework* no cenário motivador, foi utilizada uma instância do banco de dados *MySQL*, utilizada por uma instância da plataforma OpenEDX instalada em um servidor de testes[43]. Esta instância de banco de dados foi populada com um curso de entrevista motivacional que apresenta casos de entrevista motivacional para alunos de cursos de saúde e é utilizada em uma pesquisa na universidade de *Duke*[44].

O esquema lógico completo do banco de dados utilizado é apresentado no Apêndice A. Para facilitar a visualização, são consideradas as tabelas apresentadas na Figura 7.1.

[43] As configurações utilizadas no servidor são apresentadas no Apêndice B.

[44] O curso de entrevista motivacional é disponibilizado apenas para alunos da universidade. Uma cópia do curso foi utilizada nos experimentos.

Figura 7.1 – Esquema lógico das tabelas.

```
auth_user(id,username, first_name, last_name,email,password,is_staff,is_active,is_superuser)

auth_userprofile(id, user_id, name, language, location, meta, courseware, gender, mailing_address,
year_of_birth, level_of_education, goals, allow_certificate, country, city, bio,
profile_image_uploaded_at)
        user_id references auth_user

student_language_proficiency(user_profile_id, id, code)
        user_profile_id references auth_user

course (id, previous_version, root, schema_version, edited_on, original_version, edited_by,
definition)

module_definition (id, definition)

module (id, description, def)
        def references module_definition

module_child(id, id_child)
        id references module
        id_child references module

student_courseenrollment(user_id, course_id, is_active, mode, description)
        user_id references auth_user
        course_id references course

student_moduleenrollment(user_id, course_id, module_id, graded, max_grade, grade, is_active)
        user_id references auth_user
        course_id references course
        module_id references module
```

Fonte: Autoria Própria.

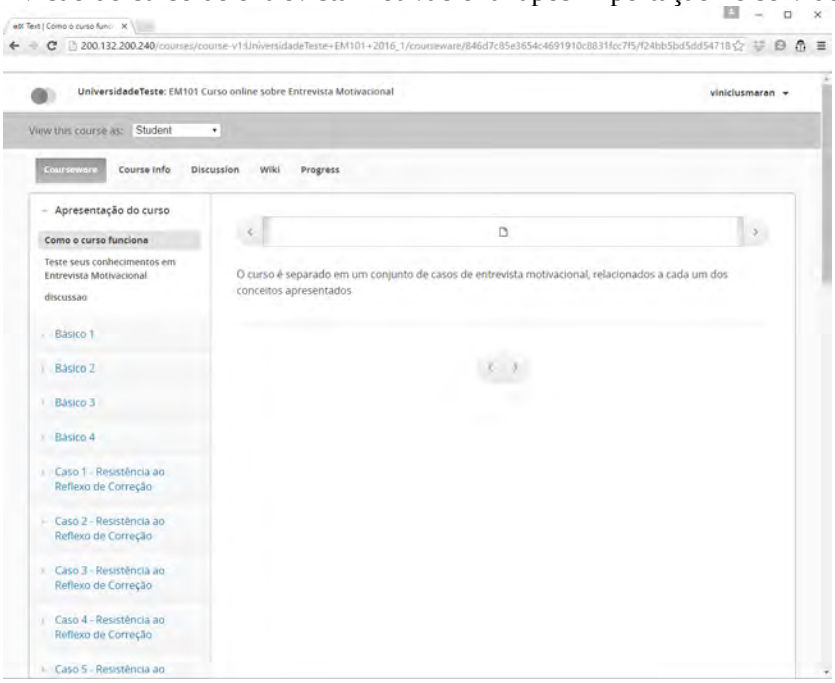
Na tabela *auth_user* são armazenadas as principais informações relacionadas ao usuário, como senha, nome e se o usuário faz parte da organização do curso. Na tabela *auth_userprofile* são armazenadas algumas informações básicas do perfil do usuário, como por exemplo o sexo, endereço de email, nível educacional, cidade, biografia, entre outros. Na tabela *student_language_proficiency* são armazenadas as proficiências do aluno em alguma linguagem.

Em relação aos cursos, os mesmos têm suas informações armazenadas na tabela *course*. Estes cursos são estruturados em módulos, que por sua vez tem suas informações armazenadas nas tabelas *module*, que armazena as informações básicas do módulo, e *module_definition*, que armazena o código em JSON do módulo. As tabelas *student_courseenrollment* e *student_moduleenrollment* armazenam as informações de ligação entre o estudante e os módulos e cursos nos quais o estudante está registrado.

Após a instalação da instância da plataforma *OpenEDX* no servidor de testes, foram realizados dois processos: (*i*) foram importadas as informações dos módulos utilizados em um curso sobre entrevista motivacional previamente criados e utilizados e, (*ii*) foram criados usuários fictícios que representam alunos, tutores e professores no MOOC.

Os módulos utilizados no curso de entrevista motivacional foram importados para o servidor de testes utilizando um processo existente na própria plataforma OpenEDX. Uma visão geral do MOOC após a importação dos módulos é apresentada na Figura 7.2.

Figura 7.2 – Visão do curso de entrevista motivacional após importação no servidor de testes.



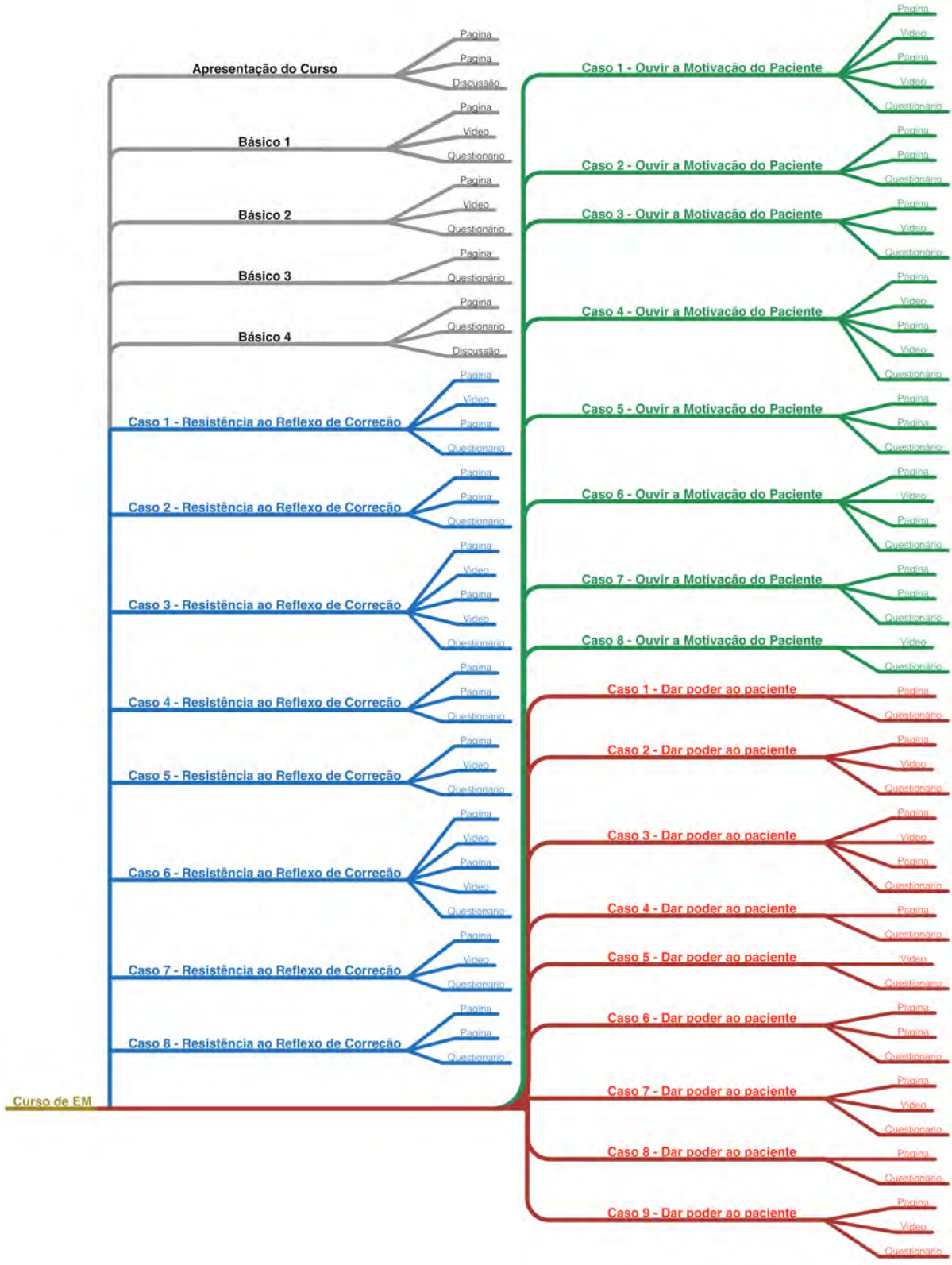
Fonte: Autoria Própria.

A Figura 7.3 apresenta a estrutura do curso utilizado nos testes, com seus respectivos módulos e recursos associados a estes módulos. O curso é composto por um conjunto de módulos. Obrigatoriamente, o aluno deve completar todos os questionários dos primeiros cinco módulos, que representam os módulos que apresentam os conceitos básicos de entrevista motivacional.

Além disso, são apresentados mais vinte e cinco casos de entrevista motivacional, onde cada um destes casos é direcionado a uma das seguintes técnicas: (*i*) Resistência ao reflexo de correção, (*ii*) Ouvir a motivação do paciente e (*iii*) Dar poder ao paciente. Além disso, cada caso pode utilizar uma das técnicas aplicadas a: (*i*) Aprendizado da técnica de forma geral, (*ii*) Aprendizado da técnica aplicada à promoção de exercício físico ou (*iii*) Aprendizado da técnica aplicada ao desejo de parar de fumar. Cada um dos módulos utilizados no curso utiliza um conjunto de recursos associados ao módulo. Estes recursos podem ser: (i) vídeo, (ii) página web, (iii) fórum de discussão ou (iv) questionário.

Além da utilização da instância de banco de dados com as informações sobre o curso de entrevista motivacional, foi utilizada uma ontologia que descreve informações de contexto gerenciadas pelo middleware ubíquo.

Figura 7.3 – Estrutura do curso de EM utilizada nos testes.

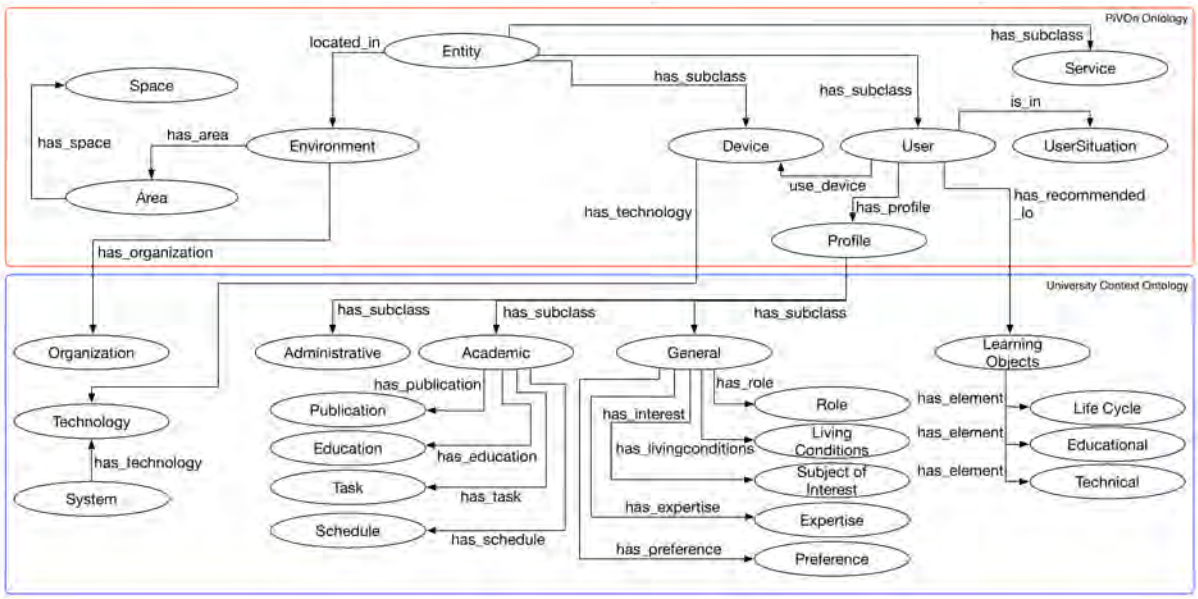


Fonte: Autoria Própria.

A definição desta ontologia foi realizada baseando-se nos conceitos definidos na ontologia PiVOn e em uma implementação recente de recomendação de recursos em um

ambiente universitário (MACHADO; PALAZZO, 2014). Uma visão geral da rede de ontologias de contexto utilizada nos testes é apresentada na Figura 7.4.

Figura 7.4 – Rede de ontologias utilizada para representar contextos de interesse dos estudantes.



Fonte: Autoria Própria. Baseada nas definições de (MACHADO; PALAZZO, 2014).

A rede de ontologias proposta por Machado & Palazzo (2014) representou informações de contexto de um ambiente universitário e permitiu a realização de inferências de recomendação de materiais de apoio e de objetos de aprendizagem. A rede de ontologias representou informações de contexto referentes a quatro dimensões de contexto: (*i*) Localização; (*ii*) Tecnologia, (*iii*) Perfil e (*iv*) Objetos de Aprendizagem.

Após a definição das fontes de informação utilizadas na avaliação, o *framework* foi aplicado no cenário de aplicação apresentado anteriormente. O processo de aplicação do *framework* com a utilização da metodologia é apresentado a seguir.

7.3 Aplicação do *Framework* no Cenário e Avaliação

Para realizar a avaliação do *framework*, o mesmo foi aplicado ao cenário de uso apresentado anteriormente com suporte do protótipo da arquitetura. A aplicação do *framework* foi realizada através da execução dos passos descritos na metodologia apresentada nesta tese. A Figura 7.5 apresenta uma visão geral da metodologia, separada por *workflows*, e o nível de envolvimento de cada especialista na realização das tarefas.

A execução das tarefas é apresentada a seguir, em subseções relacionadas a cada *workflow*, com a descrição relacionada a cada etapa da figura e sua numeração correspondente na Figura 7.5.

7.3.1 *Context Workflow*

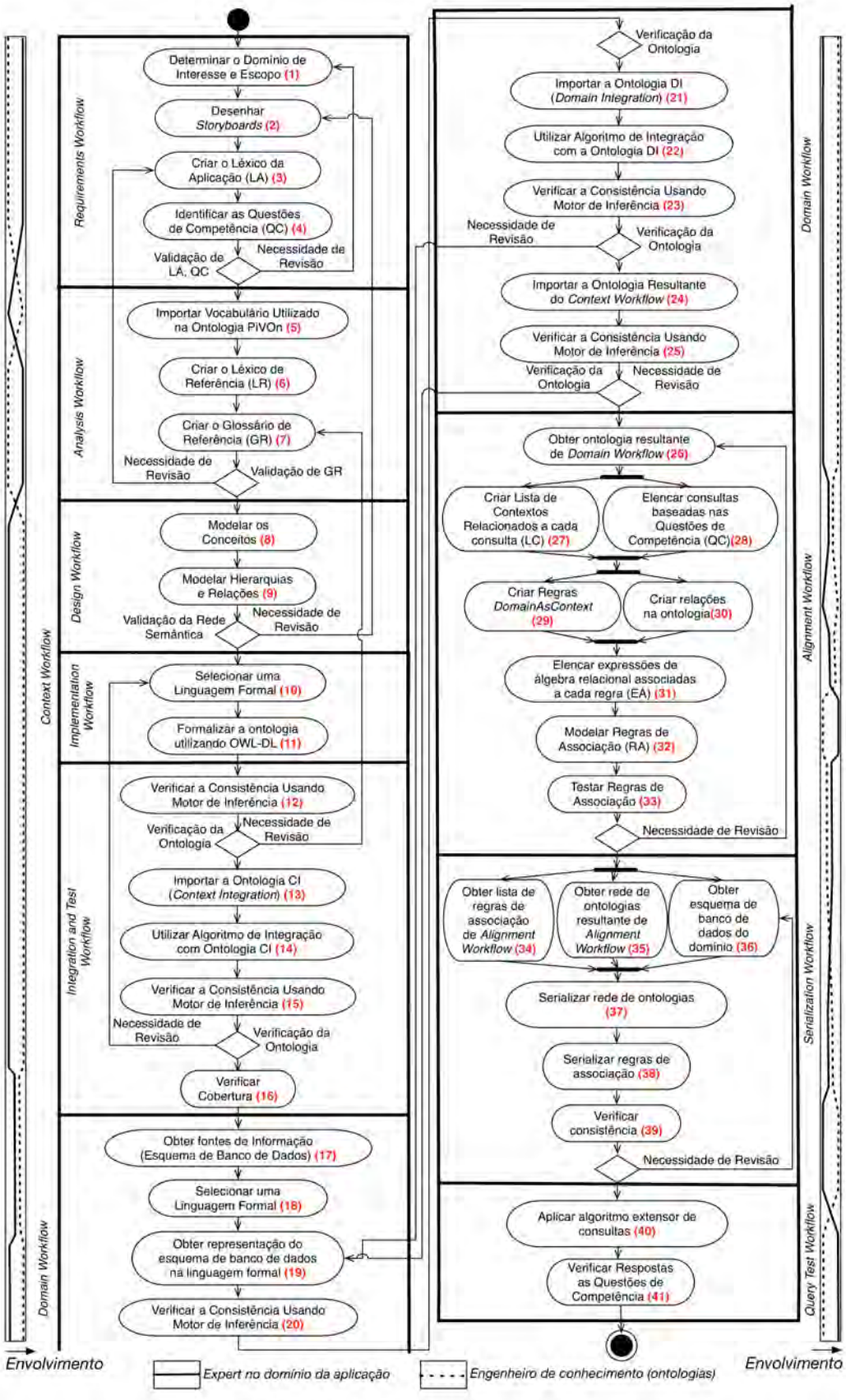
A primeira etapa no processo de integração é determinar o domínio de interesse e o escopo da definição da ontologia de contexto estendida (1) a partir do uso da ontologia utilizada no *middleware* ubíquo. Considerando o cenário de aplicação apresentado anteriormente na seção 7.1, o domínio de interesse da ontologia de contexto utilizada pelo *framework* é *a apresentação de módulos do curso de entrevista motivacional de acordo com informações de contexto provenientes da ontologia utilizada pelo middleware ubíquo*. O escopo é definido como *seleção contextualizada de módulos e materiais de apoio de acordo com situações específicas de alunos, informações de contexto relacionadas a perfis, dispositivos, localização, interesses dos estudantes e foco de aprendizado*.

A partir da definição do domínio de interesse e do escopo, foi definido um conjunto de *storyboards* (2) que descrevem as situações de utilização do sistema, com o contexto envolvido em cada uma destas situações. Os *storyboards* foram extraídos da descrição do cenário de aplicação.

***Storyboard* 1**: *João é aluno do curso de saúde pública no campus universitário. Ele se registra e acessa o MOOC de entrevista motivacional durante o intervalo de uma das aulas através de seu smartphone enquanto está no corredor. Como João se registrou no curso e faz o primeiro acesso, ele é apresentado somente aos módulos do curso relacionados a aprendizagem de conceitos básicos de entrevista motivacional. Como João está no intervalo das aulas, utiliza um dispositivo com capacidades de visualização limitadas e o intervalo tem 15 minutos, nenhum material de apoio que possui vídeos com duração maior que 10 minutos é apresentado*;

***Storyboard* 2**: *João é aluno do curso de saúde pública no campus universitário. Ele se registrou no MOOC sobre entrevista motivacional, e durante uma das aulas no seu curso de graduação, ele apresenta o MOOC para alguns de seus colegas. Como João está em aula, o MOOC apresenta a ele somente informações gerais sobre o curso, como a apresentação sobre o curso e o fórum de discussões;*

Figura 7.5 – Visão geral da metodologia de aplicação do *framework*.



Fonte: Autoria Própria.

***Storyboard* 3**: *João é aluno do curso de saúde pública no campus universitário. João tem interesse em áreas como promoção de atividades físicas para pacientes e combate ao tabagismo. Este interesse se deve a dois fatores principais: (i) o pai de João fuma, e (ii) João possui um amigo próximo com obesidade comportamental. João vai ao laboratório de informática após o almoço e realiza o acesso ao MOOC de entrevista motivacional através de um dos computadores do laboratório. João já completou as atividades referentes ao aprendizado de conceitos básicos de entrevista motivacional. Assim, são apresentados a João as partes do curso relacionadas aos conceitos de "ouvir a motivação do paciente", "resistência ao reflexo de correção" e "dar poder ao paciente". São apresentados apenas os casos onde o foco é a promoção de exercícios físicos ou ao combate ao tabagismo.*

Após a definição dos *storyboards*, foi definido o Léxico da Aplicação (3), apresentado na Tabela 7.1. O léxico de aplicação foi definido com base em consultas sobre os conceitos relacionados ao curso de EM e a ontologia de contexto utilizada.

Tabela 7.1 – Léxico da aplicação

Aluno	Curso	Saúde Pública
MOOC	Entrevista Motivacional	Intervalo da Aula
Prédio	Sala de Aula	Aula
Laboratório	Dispositivo	Celular
Computador Desktop	Vídeo	Questionário
Parte do Curso	*Smartphone*	Duração
Saúde Pública	Campus Universitário	Corredor
Apresentação	Fórum de Discussão	Promoção de Atividade Física
Combate ao Tabagismo	Conceitos Básicos	Ouvir a motivação do paciente
Resistência ao Reflexo de Correção	Dar poder ao Paciente	Material de Apoio

Fonte: Autoria Própria.

Baseado nos *storyboards* previamente criados e na definição do léxico da aplicação, foram definidas as questões de competência (4) que o *framework*, após a aplicação das regras de ligação, deve ser capaz de responder. O conjunto de questões de competência é apresentado na Tabela 7.2.

Após a definição das questões de competência, o vocabulário da ontologia de contexto foi importado (5). O vocabulário da ontologia de contexto é composto pelos nomes das classes, propriedades de objeto e propriedades de dado representadas. A Tabela 7.3 apresenta os termos que compõem o vocabulário importado da ontologia[45].

[45] O número de termos foi limitado em 15 por questões de espaço. O vocabulário completo é apresentado no Apêndice C.

Tabela 7.2 – Questões de competência

QC1	Quais materiais de apoio do curso de entrevista motivacional devem ser apresentados aos alunos quando eles estão no intervalo de aula ?
QC2	Quais materiais de apoio do curso de entrevista motivacional devem ser apresentados aos alunos quando eles estão acessando o curso utilizando *smatphones* ?
QC3	Quais casos do curso de entrevista motivacional devem ser apresentados aos alunos que possuem interesse nos temas “combate ao tabagismo” e “incentivo ao exercício físico”?
QC4	Quais casos do curso devem ser apresentados ao aluno quando ele estiver acessando o curso durante uma aula presencial ?
QC5	Quais casos do curso de entrevista motivacional devem ser apresentados ao aluno nos primeiros acessos ao curso ?

Fonte: Autoria Própria.

Tabela 7.3 – Vocabulário importado da ontologia de contexto.

Ability	*Activity*	*Contact*
Device	*Entity*	*Publication*
System	*User*	*Work*
Role	*LearningObject*	*Expertise*
GPS	*Organization*	*Academic*

Fonte: Autoria Própria.

A próxima etapa na execução da metodologia é a criação do léxico de referência (6). O léxico de referência é gerado pela intersecção do léxico da aplicação com o vocabulário da ontologia de contexto. A partir do léxico de referência, foi definido o glossário de referência (7), com uma descrição sobre o significado de cada um dos termos. O glossário de referência gerado é apresentado na Tabela 7.4.

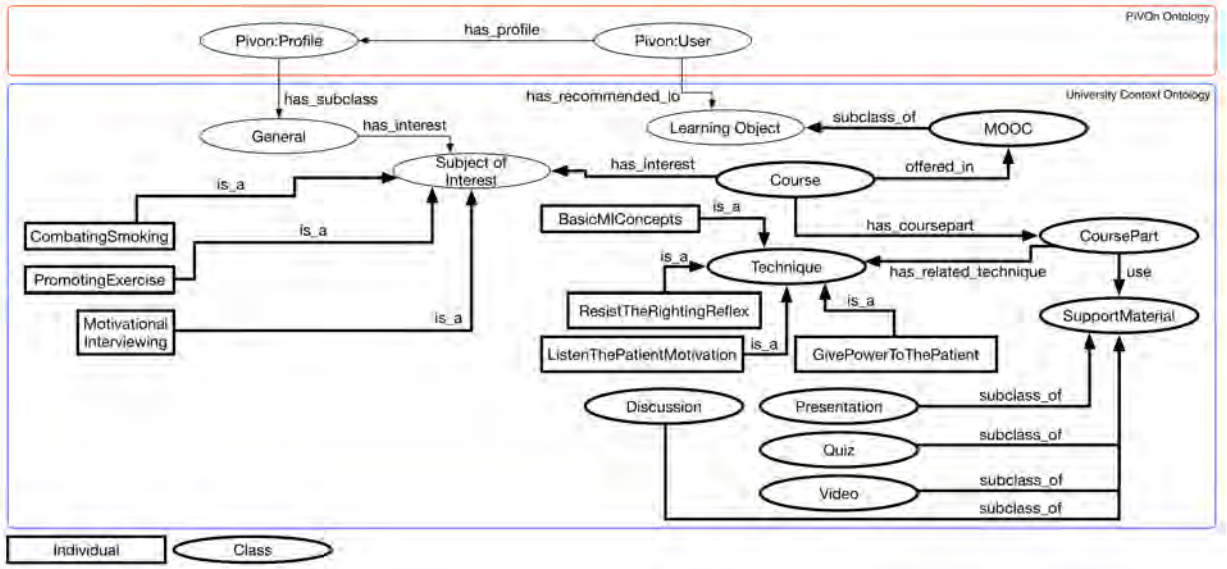
O glossário de referência gerado foi utilizado na modelagem de conceitos (8) e relações (9) na rede de ontologias. Cada um dos termos do glossário foi classificado conforme seu tipo de informação e foi relacionado aos termos já definidos na ontologia de contexto. Na Figura 7.6 é apresentada a rede semântica de conceitos. Como pode ser observado na figura, alguns dos conceitos foram modelados como classes, enquanto outros conceitos foram modelados como indivíduos na rede semântica. Além disso, relações foram criadas entre as novas definições e as definições já existentes na ontologia de contexto. As definições criadas neste processo foram marcadas na Figura 7.6.

Tabela 7.4 – Glossário de referência

Termo	Significado
MOOC	Cursos *online* disponibilizados para uma grande audiência, que geralmente não é limitada geograficamente
Curso	Curso oferecido em uma plataforma de MOOCs
Parte do Curso	Uma parte do curso oferecido em uma plataforma de MOOCs. Partes de cursos são compostas de elementos de apoio definidos na plataforma
Entrevista Motivacional	Tema principal do curso utilizado no cenário de aplicação
Conceitos Básicos	Foco de aprendizagem do aluno sobre o curso durante os primeiros acessos
Combate ao Tabagismo	Assunto de interesse do aluno sobre casos de entrevista motivacional que apresentam situações de combate ao tabagismo
Promoção de Atividade Física	Assunto de interesse do aluno sobre casos de entrevista motivacional que apresentam situações promoção de atividade física
Resistência ao Reflexo de Correção	Uma das técnicas utilizadas em entrevistas motivacionais
Ouvir a motivação do paciente	Uma das técnicas utilizadas em entrevistas motivacionais
Dar poder ao Paciente	Uma das técnicas utilizadas em entrevistas motivacionais
Apresentação	Um dos tipos de material de apoio associado a uma parte do curso na plataforma de MOOCs
Vídeo	Um dos tipos de material de apoio associado a uma parte do curso na plataforma de MOOCs
Fórum de Discussão	Um dos tipos de material de apoio associado a uma parte do curso na plataforma de MOOCs
Questionário	Um dos tipos de material de apoio associado a uma parte do curso na plataforma de MOOCs

Fonte: Autoria Própria.

Figura 7.6 – Rede semântica gerada no *Design Workflow*.



Fonte: Autoria Própria.

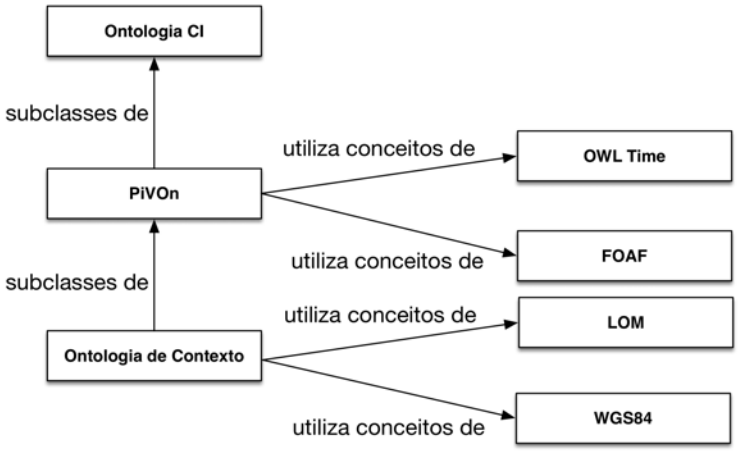
A linguagem de representação escolhida (10) para a representação da rede semântica foi a linguagem OWL-DL, para manter a compatibilidade com as representações utilizadas

pela ontologia de contexto. A rede de ontologias foi formalizada através do *software* Protégé (11). Após a formalização da rede de ontologias, foi verificada a consistência da rede de ontologias com a utilização do motor de inferência Pellet (12).

A ontologia CI foi importada (13) e integrada (14) à rede de ontologias gerada no processo anterior com a utilização do *plugin* SQL-eCO. A rede de ontologias resultante do processo é apresentada na Figura 7.7[46].

A rede de ontologias representada em um arquivo OWL-DL teve a sua consistência verificada com o motor de inferência Pellet (15). Esta rede de ontologias é utilizada nas próximas etapas da metodologia. A Tabela 7.5 apresenta uma visão geral da rede de ontologias considerando o número de definições.

Figura 7.7 – Rede de ontologias gerada no processo de integração de *Context Workflow*.



Fonte: Autoria Própria.

Tabela 7.5 – Visão geral da rede de ontologias gerada para representação de contexto.

Ontologia	**Classes**	**Propriedades de Objeto**	**Propriedades de Dado**	**Axiomas**
Ontologia CI	2	3	0	7
PiVOn (com importação de OWLTime e FOAF)	88	113	75	1061
Rede de ontologias completa (após aplicação do *workflow*)	201	192	176	2683

Fonte: Autoria Própria.

A verificação de cobertura da ontologia (16) é feita de forma subjetiva pelo *expert* no domínio de aplicação e pelo engenheiro de ontologias. Esta verificação é realizada através da verificação se todos os contextos de interesse foram modelados em conceitos na ontologia.

Esta verificação foi realizada. Após a definição da rede de ontologias de representação de contexto, o *workflow* relacionado ao domínio de aplicação foi executado. O processo de execução do *workflow* é apresentado na próxima seção.

7.3.2 *Domain Workflow*

Para obter o acesso ao esquema de banco de dados (17), foi configurado um acesso ao banco de dados utilizado na plataforma *OpenEDX* através do software *MySQL Workbench*[47] utilizando uma conexão SSH. A linguagem escolhida para a fornalização do esquema (18) foi a linguagem OWL-DL, para manter a compatibilidade com a ontologia de contexto utilizada pelo *middleware* ubíquo.

Após, foi obtida a representação em OWL-DL do esquema de banco de dados (19). A representação foi obtida com a utilização da ferramenta RDBToOnto, com as configurações de conversão previamente apresentadas no capítulo 5. A Tabela 7.6 apresenta uma visão geral, em números de elementos, da representação em OWL-DL resultante do processo de conversão.

Tabela 7.6 – Visão geral da representação em OWL-DL do esquema de banco de dados.

Classes	**Propriedades de Objeto**	**Propriedades de Dado**	**Axiomas**
130	262	620	3654

Fonte: Autoria Própria.

A representação em OWL-DL foi validada sintaticamente e semanticamente através da abertura do arquivo OWL-DL no *software* Protégé e a verificação de consistência (20) utilizando o motor de inferência Pellet. Após a realização desta verificação, o *plugin* para a ferramenta Protégé SQL-eCO foi utilizado para fazer a importação (21) e integração (22) da ontologia DI com a representação em OWL-DL do esquema de banco de dados.

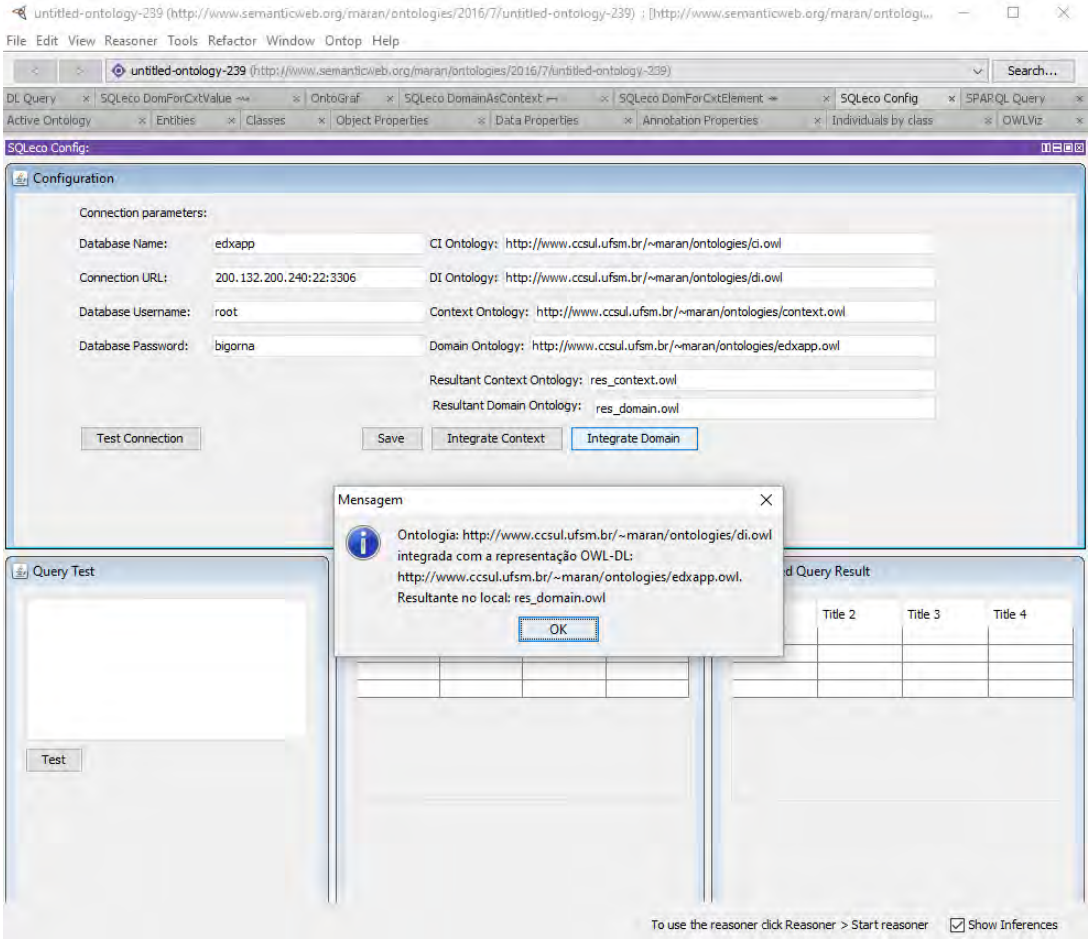
A Figura 7.8 apresenta a interface do *plugin* SQL-eCO após a realização da integração. Como saída deste processo, uma rede de ontologias é gerada e armazenada no local definido no *plugin*. Após a realização desta etapa, a consistência desta rede de ontologias foi verificada através da utilização do motor de inferência Pellet (23).

---

[46] Ontologia OWL-Time: https://www.w3.org/TR/owl-time/. Ontologia FOAF: xmlns.com/foaf/0.1/. Ontologia LOM: slor.sourceforge.net/ontology/lom.owl. Ontologia WGS84: https://www.w3.org/2003/01/geo/wgs84_pos

[47] https://www.mysql.com/products/workbench/

Figura 7.8 – Integração entre a ontologia DI e a representação OWL-DL do esquema de banco de dados no *plugin* SQL-eCO.



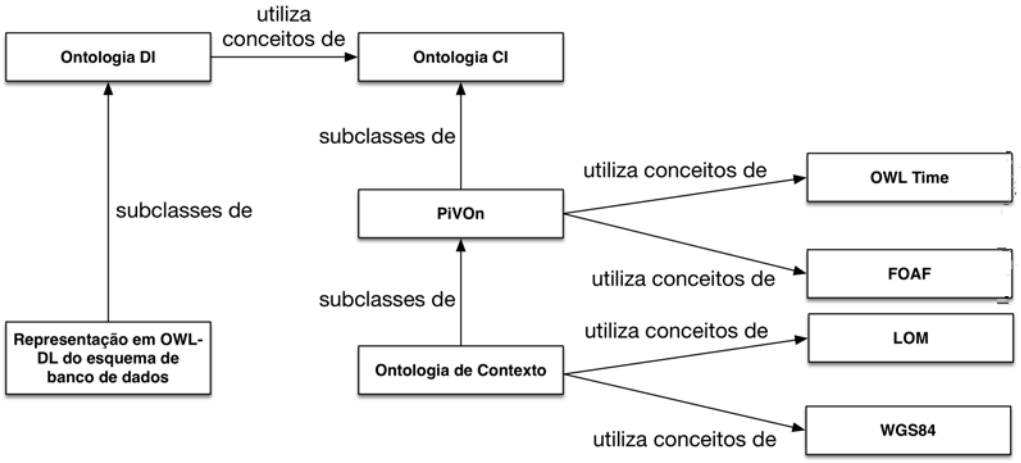
Fonte: Autoria Própria.

A rede de ontologias resultante deste processo contém apenas as representações relativas ao domínio da aplicação. Para que as regras de ligação possam ser criadas considerando uma rede de ontologias, faz-se necessária a integração da rede de ontologias resultante do *Context Workflow* (24).

Esta integração foi realizada através do *plugin* SQL-eCO, através da definição do local das redes de ontologias resultantes de *Context Workflow* e da integração entre a ontologia DI e a representação em OWL-DL do esquema de banco de dados do domínio. Como resultado deste processo, uma rede de ontologias foi gerada. Esta rede de ontologias teve sua consistência verificada através do motor de inferência Pellet (25).

A saída do processo de integração é uma rede de ontologias, representada em um arquivo OWL-DL. Uma visão geral da rede de ontologias é apresentada na Figura 7.9.

Figura 7.9 – Rede de ontologias resultante dos processos do *Domain Workflow*.



Fonte: Autoria Própria.

Esta rede de ontologias é utilizada no *Aligment Workflow* para a criação de regras de ligação. O processo de criação das regras é apresentado na próxima seção.

7.3.3 *Alignment Workflow*

A partir da rede de ontologias gerada no *workflow* anterior e o banco de dados de domínio, as regras de ligação podem ser definidas e testadas. Para isto, o arquivo que representa a rede de ontologias resultante foi aberto no *software* Protégé (26) e uma conexão com o banco de dados de domínio foi configurada no *plugin* SQL-eCO.

Para modelar as regras de ligação foi necessário elencar os elementos de contexto de interesse relacionados a cada questão de competência (27). A Tabela 7.7 apresenta os elementos de contexto relacionados a cada uma das questões de competência e sua classificação na ontologia de contexto (classe, propriedade de objeto, propriedade de dado ou indivíduo).

Tabela 7.7 – Elementos de contexto de interesse relacionados a cada questão de competência

Questão de Competência	Elementos de Contexto de Interesse	Classificação do Elemento de Contexto
QC1	*in_class_break*	Indivíduo da classe *UserSituation*
QC2	*limited_viewing = true*	Propriedade de dado da classe *Device*
QC3	*somoking_cessation, exercise_promotion*	Indivíduos da classe *SubjectOfInterest*
QC4	*in_class*	Indivíduo da classe *UserSituation*
QC5	*basic_MI_concepts, partes do curso*	Indivíduo da classe *Technique, indivíduos da classe CoursePart*

Fonte: Autoria Própria.

Além dos elementos de contexto elencados, foram elencadas as consultas relacionadas a cada questão de competência (28). As consultas elencadas nesta etapa são as consultas sobre as quais se deseja realizar a contextualização dos resultados. Estas consultas são elencadas nesta etapa da mesma forma que são utilizadas pelo sistema de informação, sem modificações. A Tabela 7.8 apresenta as consultas utilizadas para retornar as informações necessárias para a apresentação dos cursos na plataforma *OpenEDX*.

Tabela 7.8 – Consultas utilizadas no sistema relacionadas a cada questão de competência

Questão de Competência	Consulta	Id da consulta
QC1, QC2	**SELECT** t1.idmodule AS lev1, t2.idmodule AS lev2, t3.idmodule AS lev3, t4.idmodule AS lev4, def.definition AS lev4definition **FROM** module_child **AS** t1 **LEFT JOIN** module_child **AS** t2 **ON** t2.idmodule = t1.idchild **LEFT JOIN** module_child **AS t3 ON** t3.idmodule = t2.idchild **LEFT JOIN** module_child **AS t4 ON** t4.idmodule = t3.idchild **JOIN** module **ON** t4.idmodule = module.id **JOIN** module_definition **AS** def **ON** module.definition=def.id **WHERE** t1.idmodule = '57bb8878e4efae083b63c157'	Cons1
QC3, QC4, QC5	**SELECT** course.id, t1.idmodule AS lev1, t2.idmodule AS lev2, module_definition.definition **FROM** auth_user **JOIN** student_courseenrollment **ON** (auth_user.id = student_courseenrollment.user_id) **JOIN** course **ON** (student_courseenrollment.course_id = course.id) **JOIN** module_child **AS** t1 **ON** (t1.idmodule = course.module_id) **LEFT JOIN** module_child **AS** t2 **ON** (t2.idmodule = t1.idchild) **JOIN** module **ON** (t2.idmodule = module.id) **JOIN** module_definition **ON** module.definition = module_definition.id **WHERE** auth_user.id = 5 **AND** course.id='course-v1:UnivTest+EM101+2016_1'	Cons2

Fonte: Autoria Própria.

As consultas se repetem em questões de competência diferentes, pois algumas das questões de competência compartilham a mesma porção de domínio, porém variam em elementos de contexto. No cenário de aplicação, as questões de competência QC1 e QC2 utilizam a mesma consulta, que retorna a estrutura de árvore dos módulos e seus submódulos, bem como a definição em JSON destes módulos. Esta estrutura é apresentada a partir de um módulo informado pelo sistema, que é utilizado na cláusula *WHERE* da consulta.

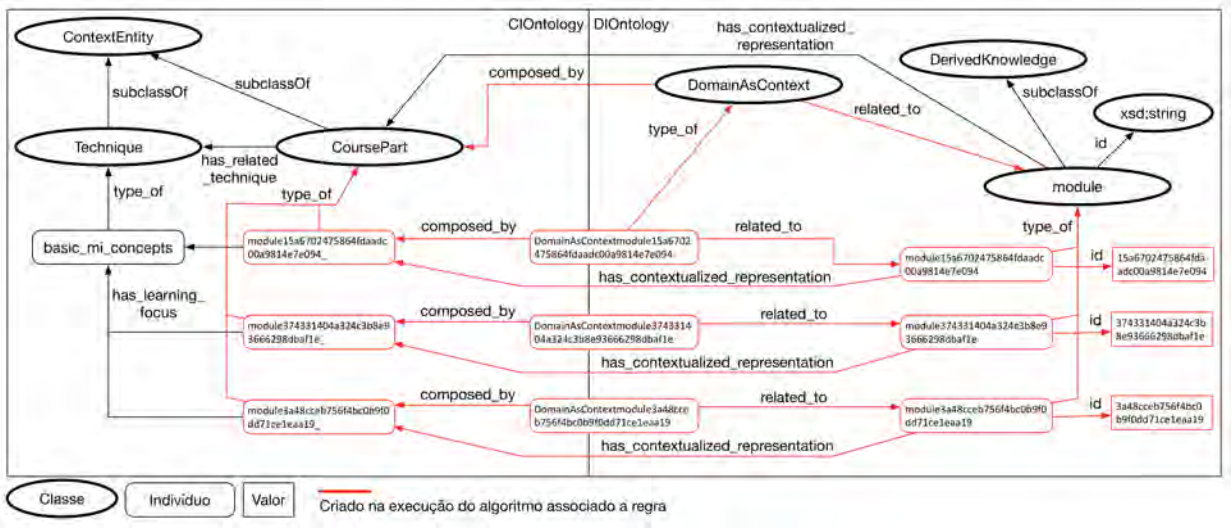
As questões QC3, QC4 e QC5 também utilizam a mesma consulta. A consulta utilizada nestas questões de competência retorna as estruturas básicas de um curso, com os principais módulos do curso realizado pelo aluno que tem seu *id* informado na cláusula *WHERE* da consulta.

Como originalmente nenhum dos módulos dos cursos estavam relacionados a alguma técnica de EM e a definição dos indivíduos da classe *Technique* foi feita durante a aplicação

da metodologia, foi necessário definir a regra *DomainAsContext* (29) para cada um dos módulos principais do curso de entrevista motivacional[48], associando-os à classe *CoursePart*. Um exemplo de regra criada é apresentado a seguir: *module(846d7c85e3654c4691910c8831fcc7f5) ↔ CoursePart.*

Após a definição das regras e aplicação das mesmas utilizando o *plugin* SQL-eCO, a rede de ontologias foi modificada com a inserção dos indivíduos que representam os módulos na ontologia de contexto[49]. Com a criação destes indivíduos, foi possível associar cada parte de curso a um indivíduo da classe *Technique*, que representa uma técnica de EM (30). A Figura 7.10 apresenta um exemplo das relações *has_interest* realizadas entre os indivíduos da classe *CoursePart* e da classe *Technique*.

Figura 7.10 – Relações entre os indivíduos das classes *CoursePart* e *Technique*.



Fonte: Autoria Própria.

Após a definição das regras do tipo *DomainAsContext* e a realização das ligações necessárias na ontologia, foram modeladas as regras de ligação relacionadas aos elementos de contexto de interesse relacionados as questões de competência. Para modelar as regras, a primeira etapa é elencar as expressões em álgebra relacional relacionadas a cada elemento de contexto de interesse (31). As expressões de consulta elencadas de acordo com cada uma das questões de competência são apresentadas na Tabela 7.9.

[48] Os módulos principais do curso de entrevista motivacional são aqueles que representam o segundo nível na árvore apresentada na Figura 7.3 (Exemplo: Caso 2 – Ouvir a motivação do paciente). No banco de dados há mais níveis na árvore que representa a estrutura do curso. Alguns níveis foram ocultados na figura que apresenta a estrutura do curso para melhorar a visualização.

[49] A criação e associação entre os indivíduos é realizada pelo algoritmo associado à regra de ligação.

Tabela 7.9 – Expressões de consulta relacionadas a cada questão de competência

Questão de Competência	Consulta	Identificador
QC1	`SELECT module.* FROM module JOIN module_definition ON (module.definition = module_definition.id) WHERE module_definition.definition->"$.block_type" <> 'video' OR module_definition.definition->"$.block_info.duration" < 600`	Consulta1
QC2	`SELECT module.* FROM module JOIN module_definition ON (module.definition = module_definition.id) WHERE module_definition.definition->"$.block_type" <> 'video'`	Consulta2
QC3	`SELECT module.* FROM module JOIN module_definition ON (module.definition = module_definition.id) WHERE module_definition.definition->"$.block_type" = 'overview' OR module_definition.definition->"$.block_type" = 'vertical' OR module_definition.definition->"$.block_type" = 'course' OR module.id = '3a48cceb756f4bc0b9f0dd71ce1eaa19' OR module.id = '3e9a630776124e99bc99981a4b059c62' OR module.id = '415a6ac9bfc4473b966373bc93dc1188' OR module.id = '4d3dc080d95847a4a2ce08f6787ae41e' OR module.id = '5a789820e1104d1faae15ce505645f4e' OR module.id = '5edf5b08dbf74eb3ae6b7d5a610f2233' OR module.id = '61f8dc1e35d74b3c9ad616bc71ad7ca8' OR module.id = '64c7fedd61d34966b22ef8274d680041' OR module.id = '67df3f7dc2564f359d74c568f8af5801' OR module.id = '6cdda1f46cc44af3a2a6a538682e93b0' OR module.id = '7463bad035114ad89998721a8219059c' OR module.id = '846d7c85e3654c4691910c8831fcc7f5' OR module.id = '846d7c85e3654c4691910c8831fcc7f5' OR module.id = '846d7c85e3654c4691910c8831fcc7f5'`	Consulta3
	`SELECT module.* FROM module JOIN module_definition ON (module.definition = module_definition.id) WHERE module_definition.definition->"$.block_type" = 'overview' OR module_definition.definition->"$.block_type" = 'vertical' OR module_definition.definition->"$.block_type" = 'course' OR module.id = '8908bc18fb424a0d8c3acda022d141ea' OR module.id = '920fcc05a4b34b909b7c8ea1207320ec' OR module.id = 'a03572cb9b2c433095934b3a806558a0' OR module.id = 'a2a04bd62fef455a960d8bea83424f60' OR module.id = 'b07ff10b96e146c5ab9a14092a769c81' OR module.id = 'd9f94df02284465c906ab4998fd22132' OR module.id = 'e5017a909b0e4ec2945a6ff7814943c8'`	Consulta4
QC4	`SELECT module.* FROM module JOIN module_definition ON (module.definition = module_definition.id) WHERE module_definition.definition->"$.block_type" = 'overview' OR module_definition.definition->"$.block_type" = 'discussion' OR module_definition.definition->"$.block_type" = 'course'`	Consulta5

Fonte: Autoria Própria.

Cada consulta possui um identificador de consulta, que é utilizado posteriormente na definição das regras de ligação[50].

[50] A criação do *identificador* visa apenas facilitar a visualização da tabela de regras de ligação.

A *Consulta1* foi definida para permitir a apresentação de todos os módulos do curso, porém limitando a apresentação dos módulos que contenham vídeos com duração menor que 10 minutos (600 segundos).

A *Consulta2* foi definida para limitar apenas a apresentação de módulos que não possuam vídeos associados, devido as capacidades limitadas de dispositivos que estão no contexto associado a questão de competência 2.

As *Consultas 3 e 4* foram definidas para apresentar os módulos que representam casos de entrevista motivacional relacionados à promoção de exercício físico e combate ao tabagismo. A *Consulta5* foi definida para permitir apenas a visualização dos módulos do tipo *course*, *vertical* ou *overview*. Assim, a estrutura básica do curso é apresentada ao aluno.

Para modelar as regras de ligação (32), foi utilizado o *plugin* SQL-eCO. As regras foram modeladas através da seleção do elemento de contexto de interesse relacionado à regra, e à definição da consulta na linguagem SQL que representa a expressão em álgebra relacional relacionada à regra. Na Tabela 7.10 são apresentadas as regras de ligação modeladas.

Tabela 7.10 – Regras de ligação correspondentes a cada questão de competência

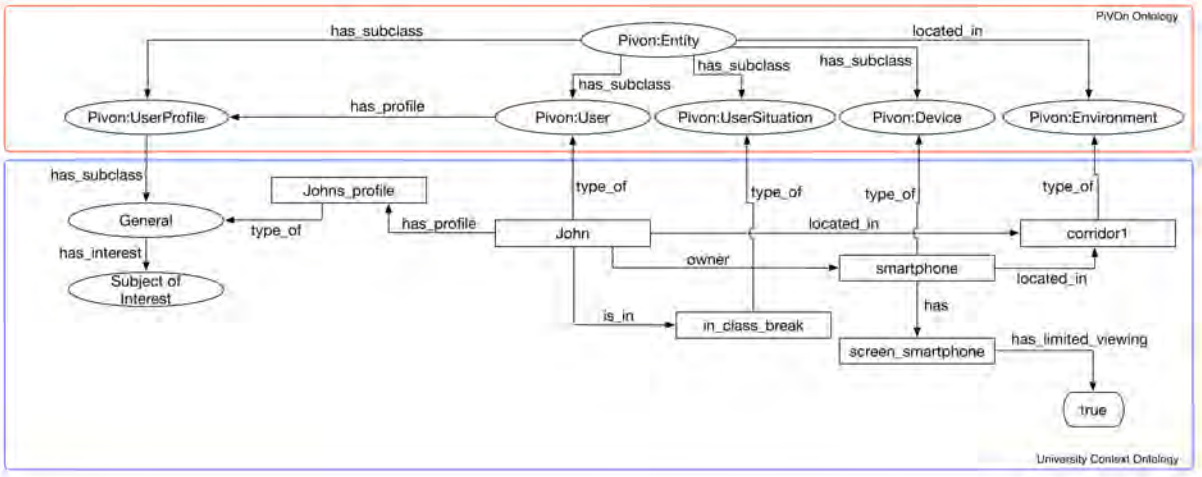
Questão de Competência	Regra de Ligação
QC1	$UserSituation_{(in\_class\_break)} \twoheadrightarrow Consulta1$
QC2	$Device_{(limited\_viewing,true,null)} \rightsquigarrow Consulta2$
QC3	$SubjectOfInterest_{(smoking\_cessation)} \twoheadrightarrow Consulta3$ $SubjectOfInterest_{(exercise\_promotion)} \twoheadrightarrow Consulta4$
QC4	$UserSituation_{(in\_class)} \twoheadrightarrow Consulta5$
QC5f	$module_{(15a6702475864fdaadc00a9814e7e094)} \leftrightarrow CoursePart$ $module_{(374331404a324c3b8e93666298dbaf1e)} \leftrightarrow CoursePart$ $module_{(bfa49b8080724351842042 63f97b273f)} \leftrightarrow CoursePart$ $module_{(f53578e5932b431886e54371c3051c3c)} \leftrightarrow CoursePart$ $module_{(f1cad38d8a264fdea3daab2b5abb993e)} \leftrightarrow CoursePart$

Fonte: Autoria Própria.

Após a modelagem das regras de ligação, as mesmas foram testadas no plugin SQL-eCO (33). Os testes relacionados às regras de ligação são realizados com a utilização de arquivos OWL-DL que representam um contexto de interesse baseado na ontologia de contexto resultante do *Context Workflow* e à realização de consultas associadas a cada um dos contextos de interesse informados.

Para realizar os testes nesta fase da metodologia, os contextos de interesse relacionados a cada um dos *storyboards* que descrevem o cenário de aplicação foram modelados na ontologia de contexto resultante do *Context Workflow*. Foram criados apenas os indivíduos na ontologia que representam os contextos de interesse relevantes para o cenário de aplicação. Os contextos de interesse foram modelados no *software* Protégé. O primeiro contexto de interesse (Figura 7.11) modelado está relacionado a situação apresentada no *storyboard 1*.

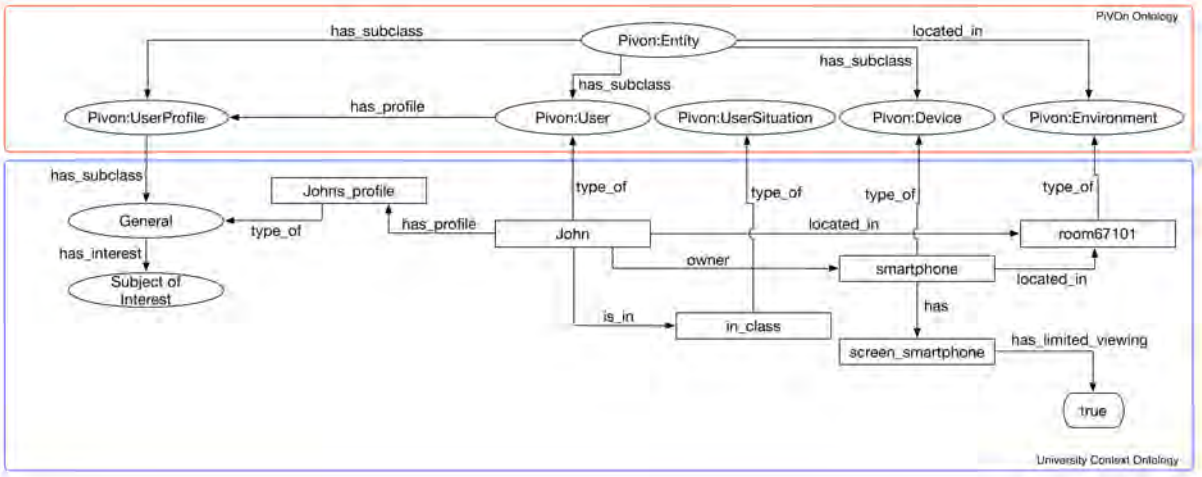
Figura 7.11 – Contexto de interesse A.



Fonte: Autoria Própria.

O segundo contexto de interesse (Figura 7.12) modelado está relacionado a situação apresentada no *storyboard 2*.

Figura 7.12 – Contexto de interesse B.

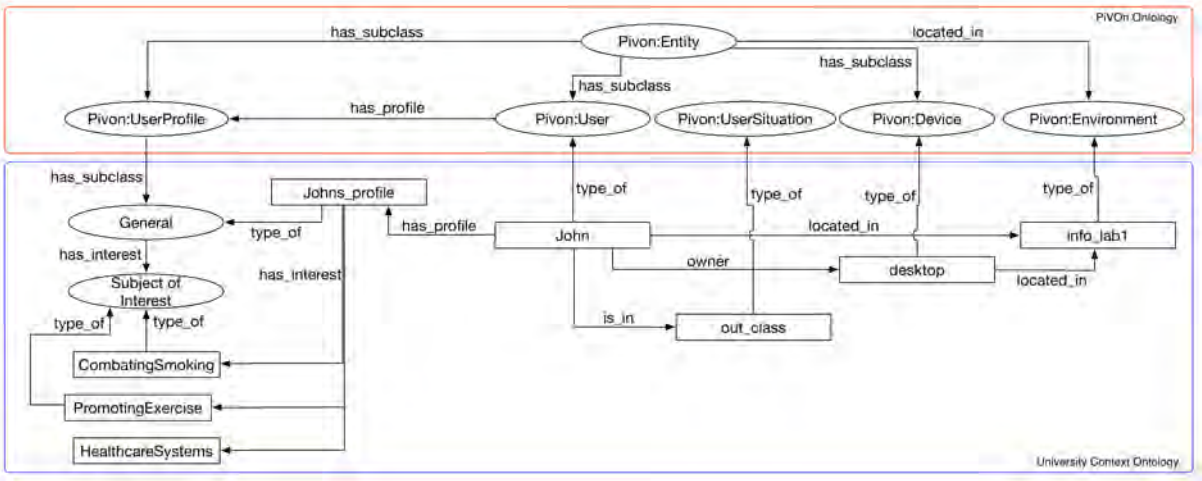


Fonte: Autoria Própria.

O terceiro contexto de interesse (Figura 7.13) modelado está relacionado a situação apresentada no *storyboard 3*.

Nesta fase, é verificada apenas a inexistência de conjuntos vazios como resultado das consultas realizadas. Para realizar esta verificação, foram testadas as consultas *Consulta1*, *Consulta2*, *Consulta3*, *Consulta4* e *Consulta5*, que representam as expressões relacionadas a cada uma das questões de competência.

Figura 7.13 – Contexto de interesse C.



Fonte: Autoria Própria.

Além disso, foram testadas as consultas *Cons1* e *Cons2*. Foram realizadas verificações com os três contextos de interesse apresentados anteriormente em conjunto com as duas consultas relacionais, e em nenhuma delas houveram conjuntos vazios como resultados das consultas. A verificação das respostas de competência é realizada posteriormente, no *Query Test Workflow*. Após a realização da verificação das regras de ligação, foi executado o *Serialization Workflow*, apresentado a seguir.

7.3.4 *Serialization Workflow*

O processo de serialização das definições utilizadas pelo *framework* é realizado com o auxílio do *plugin* SQL-eCO. Na aba de configurações, foram assinaladas as localizações das ontologias utilizadas na rede de ontologias (35) e foi configurada a conexão de banco de dados (36). Além disso, foi utilizado o arquivo que contém as definições das regras de ligação (34). Deste modo, foi realizada a serialização das definições no formato JSON (37) e

posteriormente foram persistidas no banco de dados relacional MySQL. Além disso, foram serializadas as regras de ligação (38) e persistidas na mesma instância de banco de dados.

Para verificar a consistência das definições após a persistência dos mesmos no banco de dados relacional (39), o *plugin* SQL-eCO foi utilizado, em conjunto com o motor de inferência Pellet. Não foram encontrados erros de consistência. Após a realização desta etapa, foram realizados os testes de extensão de consultas para avaliação das tuplas retornadas após a aplicação do algoritmo extensor de consultas. O processo de testes é apresentado na próxima seção.

7.3.5 *Query Test Workflow*

O *workflow* de testes e avaliação de aplicação do *framework* é realizado em duas partes: (*i*) Aplicar o algoritmo extensor de consultas (40) e (*ii*) Verificar respostas das questões de competência (41).

Para aplicar o algoritmo extensor de consultas, uma API[51] deve ser importada no projeto do sistema de informação no qual se deseja realizar a extensão de consutas. Assim, as consultas utilizadas no sistema não são realizadas diretamente através do *driver* JDBC originalmente utilizado, mas sim através da API que redireciona a consulta para o extensor de consutas. A API foi importada no protótipo que implementa um simulador de consultas. Após a realização desta etapa, foi realizada a avaliação das consultas e os resultados destas consultas após a aplicação do *framework*, variando o contexto de interesse informado.

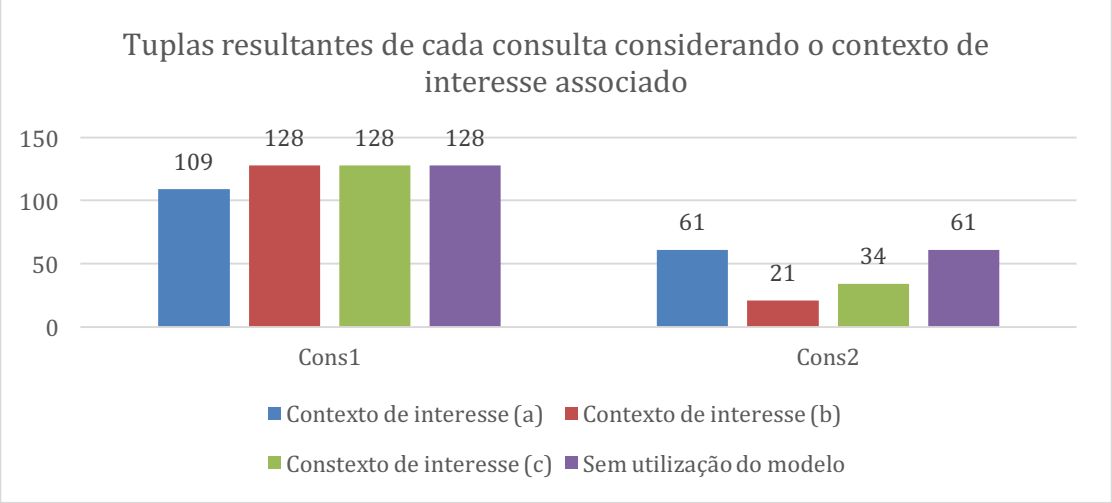
A avaliação da extensão de consultas foi realizada através da análise das tuplas retornadas nas consultas comparando os resultados destas consultas com as consultas que seriam normalmente realizadas e os resultados esperados destas consultas. Esta comparação é realizada com base nas questões de competência.

Inicialmente, foram analisadas as quantidades de tuplas retornadas nas consultas comparando as quantidades retornadas em cada consulta com e sem a utilização do *framework*, e considerando cada um dos contextos de interesse apresentados no *Alignment Workflow*, que foram modeladas utilizando como base o cenário de aplicação. A Figura 7.14 apresenta os números de tuplas resultantes em cada uma das consultas, considerando as consultas realizadas de forma tradicional, e realizadas com a utilização do *framework*, considerando cada um dos três contextos de interesse baseados no cenário de aplicação.

[51] A API importada é descrita na tese como Módulo Integrado ao Sistema de Informação.

Pode-se observar na Figura 7.14 que houve uma diminuição no número de tuplas resultantes quando o *framework* foi utilizado em conjunto com um contexto de interesse informado ao algoritmo extensor de consultas. Em relação a realização das consultas no *contexto de interesse (a)*, houve uma diminuição de 14,84% no número de tuplas retornadas na consulta *Cons1*, enquanto na consulta *Cons2* não houve mudança no número de tuplas retornadas. Isto acontece porque a extensão da consulta, neste caso, filtra apenas os módulos do curso do tipo vídeo, enquanto a consulta *Cons2* retorna apenas módulos de definição de estrutura do curso.

Figura 7.14 – Tuplas retornadas em cada consulta (*Cons1* e *Cons2*).



Fonte: Autoria Própria.

Considerando os contextos de interesse *(b)* e *(c)*, houveram mudanças apenas na consulta *Cons2*. Isto se deve ao fato de que esta consulta retorna elementos do curso que descrevem seções e casos de entrevista motivacional. A consulta *Cons1* por sua vez retorna apenas os módulos que descrevem o último nível da árvore de estrutura do curso, com os módulos que descrevem os materiais de apoio. Considerando o *contexto de interesse (b)*, houve uma diminuição de 65,57% no número de tuplas retornadas. Considerando o *contexto de interesse (c)*, houve uma diminuição de 44,26% no número de tuplas retornadas.

Além da análise quantitativa das tuplas retornadas em cada consulta considerando o contexto de interesse informado, foram analisados os resultados das consultas considerando as questões de competência definidas no *Context Workflow*. O resultado da análise é apresentado na Tabela 7.11.

Tabela 7.11 – Análise de contextos e consultas relacionadas a cada questão de competência e os resultados de cada consulta.

Questão de Competência	Consulta relacionada	Contexto de interesse utilizado	Resultado da consulta
QC1, QC2	Cons1	Contexto de interesse (a)	Todos os módulos do tipo vídeo que possuem duração maior que 600 não foram incluídos na consulta.
QC3	Cons2	Contexto de interesse (c)	Somente os módulos e submódulos do curso associados aos interesses de promoção de atividade física e controle ao tabagismo foram retornados na consulta.
QC4, QC5	Cons2	Contexto de interesse (b)	Somente os módulos do curso que apresentam a estrutura básica do mesmo (*overview* e discusões) ou que são associados à aprendizagem genérica de técnicas de EM foram retornados na consulta.

Fonte: Autoria Própria.

Pode-se obervar que os resultados das consultas testadas com os contextos de interesse derivados dos *storyboards* utilizados na metodologia de aplicação são compatíveis com as questões de competência elencadas a partir dos *storyboards*.

7.4 Conclusões do Capítulo

A aplicação do *framework* apresentado nesta tese no cenário de aplicação demonstrou a viabilidade da utilização de ontologias de alta expressividade que modelam contextos e situações em ambientes ubíquos como uma camada extra na filtragem de informações de domínio consultadas em bancos de dados relacionais, mesmo sem que a modelagem de contexto tenha sido previamente modelada pelos desenvolvedores do sistema de informação que utiliza o banco de dados de domínio.

A avaliação do *framework* foi realizada em um cenário de aplicação, com informações de contexto modeladas em uma ontologia já utilizada em trabalhos relacionados a ambientes ubíquos. Além disso, foram utilizados dados reais de um curso de EM disponibilizado em uma plataforma de MOOCs. Observou-se na aplicação do *framework* no cenário de aplicação que apesar da definição de algoritmos para tentar automatizar o processo de ligação entre definições de contexto e de domínio, ainda há a necessidade constante de acompanhamento dos processos por *experts* no domínio de aplicação e engenheiros de ontologias. Para a execução das etapas da metodologia, existe a necessidade de tomada de decisões subjetivas, a cargo do *expert* no domínio de aplicação. Esta é uma característica comum na modelagem de ontologias (DE NICOLA et al., 2009).

8 CONCLUSÕES

Nesta tese foi definido um *framework* para sistemas de informação que permite que definições de contexto altamente expressivas, representadas em ontologias, sejam utilizadas na filtragem de informações no momento da consulta destas informações em bancos de dados relacionais, mesmo sem que estes sistemas tenham sido previamente modelados como sensíveis ao contexto. Desta forma, a sensibilidade ao contexto, uma tecnologia proveniente da área de computação ubíqua, pode ser utilizada na recuperação de dados de sistemas tradicionais, sem a necessidade de reescrita de consultas, utilização de novos modelos de dados ou de novas implementações do sistema.

A área de sensibilidade ao contexto tem sido aplicada de diversas formas em sistemas ubíquos, como por exemplo na escolha de serviços para execução, adaptação de interfaces gráficas e recuperação de conteúdo. A sensibilidade ao contexto modelada em ontologias apresenta uma série de vantagens em relação a outros modelos de representação e por este motivo, tem sido constantemente utilizada em trabalhos relacionados à computação ubíqua.

Esta tese abordou questões referentes à integração de ontologias e bancos de dados e ao uso de sensibilidade ao contexto e à situação na recuperação de dados modelados de forma relacional. No Capitulo 4 foi realizada uma análise de trabalhos relacionados à proposta desta tese com uma avaliação destes trabalhos utilizando uma lista de características elencados de trabalhos relacionados e do cenário motivador de pesquisa apresentado no Capitulo 2.

Nesta análise, foi possível identificar quais características são utilizadas pelos trabalhos relacionados. A tabela 8.1 apresenta a comparação com as características do *framework* proposto nesta tese. Através da avaliação realizada neste capítulo, foi possível identificar características do *framework* em relação a três áreas: (*i*) Contexto, (*ii*) Domínio e (*iii*) Integração de Informações.

(i) **Contexto**: As definições do *framework* preveem o uso de ontologias modeladas na linguagem OWL-DL para representação de contexto, com expressividade *SHOIN(D)*. A metodologia de utilização do *framework*, bem como a avaliação realizada sobre o *framework* definem ontologias de contexto utilizadas em ambientes ubíquos. A aplicação do *framework* no cenário de aplicação apresentado demonstrou a viabilidade de utilização do *framework* na filtragem de informações baseando-se em definições de contexto ou situação;

Tabela 8.1 – Análise das características do *framework* apresentado na tese e trabalhos relacionados.

Característica / Trabalho		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Tese**
Contexto	Nível de expressividade	-	√	-	√	-	√	-	√	√
	Utilizado em arquiteturas ubíquas	-	-	-	-	-	-	-	-	√
	Modelagem de Situação	-	-	-	-	-	-	-	-	√
Domínio	Modelo Relacional	-	-	√	-	√	-	√	-	√
	Linguagem de consulta padronizada para bancos relacionais	-	-	√	-	√	-	√	-	√
Integração	Mantém consultas e esquema do banco originalmente como foram pré definidos	-	-	-	√	-	-	-	√	-
	Automatização do Processo	Parcial	-	-	√	√	√	-	√	Parcial

Fonte: Autoria Própria.

(ii) **Domínio**: O *framework*, seus componentes e definições foram formalizados considerando informações de domínio modeladas em bancos de dados relacionais. Para realizar a consulta de informações, o *framework* utiliza a linguagem SQL no processo de extensão de consultas e na definição das regras de ligação;

(iii) **Integração de Informações**: Os testes de consulta realizados no *framework* e a própria definição do *framework* utilizam consultas originalmente feitas diretamente no banco de dados de domínio. O esquema do banco de dados foi modificado apenas em relação à criação de tabelas auxiliares para armazenamento da ontologia, sem alteração nas tabelas previamente utilizadas no banco de dados. O *framework* possui algoritmos associados a cada regra de ligação. Foram apresentados ainda algoritmos que realizam o alinhamento de conceitos na rede de ontologias utilizada pelo *framework*.

A aplicação do *framework* no cenário de aplicação permitiu validar a hipótese desta tese, comprovando que *Um framework de consulta suportado por rede de ontologias pode ser utilizado para integrar informações de domínio e contexto para estender uma consulta relacional.*

O desenvolvimento da pesquisa consistiu no estudo de formas de ligação de contexto e dados de domínio, para permitir que modelagens de contexto, utilizadas em sistemas ubíquos,

sejam utilizadas como uma camada extra de filtragem na consulta de dados relacionais feitas por sistemas de informação. Com a modelagem do *framework* e metodologia propostos nesta tese, teve-se o intuito de investigar a questão de pesquisa definida no Capítulo 1.

Com a definição do *framework* e aplicação do mesmo em um cenário de aplicação através da metodologia também proposta nesta tese é possível afirmar que uma ontologia que descreve ambientes ubíquos pode ser utilizada para filtrar dados recuperados através de uma consulta em um banco de dados relacional, mesmo sem que o processo de modelagem do banco de dados tenha utilizado a modelagem de contexto baseada em ontologias no processo de definição do esquema do banco de dados. A rede de ontologias gerenciada pelos algoritmos de integração e pelos algoritmos associados às regras de integração apresentadas nesta pesquisa permitem integrar definições de contexto e de domínio. Esta integração é utilizada pelo algoritmo extensor de consultas, fazendo com que consultas feitas por um sistema de informação sejam estendidas de acordo com o contexto informado por um *middleware* ubíquo. Com a aplicação deste *framework* no cenário de aplicação apresentado no Capitulo 7, é possível concluir que a questão de pesquisa foi respondida.

A realização da pesquisa relacionada à tese possibilitou concluir que (*i*) esta pesquisa envolveu uma série de áreas relacionadas, onde termos destas áreas foram utilizados na pesquisa, como por exemplo as áreas de computação ubíqua, sensibilidade ao contexto, modelagem conceitual e de ontologias, engenharia de software e recuperação de informação; (*ii*) os estudos relacionados a sensibilidade ao contexto modelado em ontologias tem evoluído em relação a expressividade dos modelos em relação a representação de atividades e elementos de contexto, porém a adoção efetiva de modelagens de contexto baseada em ontologias em sistemas reais é limitada devido a complexidade de integração com outras soluções utilizadas em sistemas e a baixa performance de motores de inferência devido a complexidade destes modelos; (*iii*) o *framework* apresentado nesta tese, apesar de especificar algoritmos associados a cada uma das regras de ligação e algoritmos para realizar o alinhamento de conceitos na rede de ontologias, ainda depende em muitas etapas de avaliações subjetivas do *expert* no domínio de aplicação e de engenheiros de ontologias. Esta característica afeta diretamente os resultados obtidos após a aplicação do *framework* sob as consultas.

8.1 Síntese de Contribuições

As principais contribuições desta tese são:

(i) Definição de uma rede de ontologias que alinha conceitos relacionados à representação de contexto em um campus universitário inteligente e elementos de um curso de entrevista motivacional;

(ii) Definição de um *framework* que utiliza rede de ontologias para integração entre definições de contexto e de domínio e que a modelagem de contexto baseada em ontologias seja utiliza como camada de filtragem na consulta de dados relacionais;

(iii) Definição de uma metodologia de aplicação do *framework*, baseada em uma metodologia de construção de ontologias;

(iv) Implementação de uma arquitetura de software que suporta o *framework* proposto.

8.2 Trabalhos Futuros

Os seguintes trabalhos futuros poderão ser desenvolvidos baseando-se nas definições apresentadas nesta tese:

- Propor extensões das regras de ligação, com possibilidade de associação de prioridades sobre elementos de contexto a fim de realizar uma comparação mais completa em relação à sobreposição de regras utilizadas no mesmo contexto de interesse;

- Propor extensões das regras de ligação, com possibilidade de associação entre elementos de contexto utilizando operadores relacionais e lógicos de forma explícita entre elementos de contexto;

- Avaliar a aplicação da metodologia proposta nesta tese com outras ontologias de contexto e consequentemente a extensibilidade da mesma no que se refere a informações de contexto;

- Avaliar a aplicação da metodologia com bancos de dados de domínios distintos, a fim de avaliar a generalização da metodologia em relação ao domínio de aplicação;

- Verificar a possibilidade da utilização do *framework* apresentado nesta tese com soluções de sincronização entre esquemas de banco de dados relacionais e ontologias que

atualizem a ontologia que descreve o esquema de banco de dados automaticamente a cada mudança no esquema, através de mapeamentos[52].

- Avaliar a utilização de outras soluções de bancos de dados na arquitetura, com implementação e implantação da arquitetura em um sistema de informação em utilização[53];

- Avaliar a serialização e recuperação das definições das regras e ontologias utilizando SGBDs de outros modelos, além do relacional, como por exemplo utilizando soluções *NoSQL*.

8.3 Publicações

Esta seção apresenta as produções científicas que divulgaram o trabalho relacionado a esta tese, em ordem cronológica. O primeiro artigo publicado apresentou um *survey* relacionado à sensibilidade a contexto e a situações em sistemas ubíquos, técnicas de modelagem, formas de utilização destas informações, métodos de integração entre informações de domínio e de contexto e apresentou uma lista de oportunidades de pesquisa na área (relacionado as discussões apresentadas no capítulo 3).

- MARAN, Vinícius; DE OLIVEIRA, José Palazzo M. **Uma Revisão de Técnicas de Distribuição e Persistência de Informações de Contexto e Inferências de Situações em Sistemas Ubíquos**. Cadernos de Informática da UFRGS, v. 8, n. 1, p. 1-46. 2014.

No segundo artigo publicado foi apresentado um estudo relacionado a definição de requisitos importantes para arquiteturas de gerenciamento de contexto e foi realizado um comparativo entre estes requisitos e ferramentas, para verificar se as mesmas atendem aos principais requisitos de persistência de contexto para sistemas ubíquos (relacionado as discussões apresentadas nos capítulos 3 e 4).

- MARAN, Vinícius; AUGUSTIN, Iara; DE OLIVEIRA, José Palazzo M. **Are The Integrations Between Ontologies and Databases Really Opening the Closed World in Ubiquitous Computing?** In: Twenty-Sixth International Conference on Software Engineering and Knowledge Engineering (SEKE 2014). 2014. p. 453-458. Classificação Qualis: B1.

[52] Exemplo de ferramenta desenvolvida com este propósito: http://ontop.inf.unibz.it

[53] Este trabalho encontra-se em desenvolvimento pelo mestrando Diones de Dutra Vargas (UFSM - Santa Maria).

No terceiro artigo publicado foi apresentada a definição da primeira versão do método de integração entre informações de contexto e informações de domínio. Nesta primeira versão as definições de integração não estavam ainda formalizadas e foram aplicadas especificamente no domínio de *Ambient Assisted Living*, considerando a modelagem de contexto utilizada na tese de doutorado apresentada por Alencar Machado (relacionado as discussões apresentadas nos capítulos 3 e 4).

- MARAN, Vinícius; MACHADO, Alencar; AUGUSTIN, Iara; KRUG WIVES, Leandro; DE OLIVEIRA, José Palazzo M. **Proactive Domain Data Querying based on Context Information in Ambient Assisted Living Environments**. In Proceedings of the 17th International Conference on Enterprise Information Systems (ICEIS 2015), ISBN 978-989-758-097-0, p. 610-617. DOI: 10.5220/0005365006100617. Classificação Qualis: B1.

No quarto artigo publicado foi apresentada a rede de ontologias estendida a partir da ontologia PiVOn para representar contexto relacionado a MOOCs aplicados na área de educação na saúde. No artigo são apresentados (i) o processo de criação da rede de ontologias, (ii) validação da ontologia através de métodos definidos na metodologia UPON, e (iii) definição de regras para associação de materiais de apoio a contextos de interesse através da definição de regras SWRL (relacionado as discussões apresentadas nos capítulos 2 e 6).

- MARAN, Vinícius; PIETROBON, Ricardo; AUGUSTIN, Iara; DE OLIVEIRA, José Palazzo M. **Ontology Network Definition for Motivational Interviewing Learning Driven by Semantic Context-Awareness**. In: 2015 IEEE 28th International Symposium on Computer Based Medical Systems (CBMS), 2015, Sao Carlos. p. 264-272. DOI: http://dx.doi.org/10.1109/CBMS.2015.22. Classificação Qualis: B1.

No quinto artigo foi apresentada uma revisão sistemática sobre a área de computação ubíqua, especificamente em relação as áreas de sensibilidade ao contexto, adaptação e recomendação. O trabalho foi desenvolvido para verificar a associação entre as áreas de pesquisa desta tese e da tese desenvolvida por Guilherme Medeiros Machado (relacionados as discussões apresentadas no capítulo 3).

- MACHADO, Guilherme Medeiros; MARAN, Vinícius; DE OLIVEIRA, José Palazzo M.; GASPARINI, Isabela; PERNAS, Ana. **Uma Revisão Sistemática sobre as Abordagens Ubíquas para Recomendação Educacional: Estariam Elas se Tornando Adaptativas?** In: XXVI Simpósio Brasileiro de Informática na Educação, 2015, Maceió. p. 170-180. Classificação Qualis: B2.

No sexto artigo foi apresentada um estudo sobre modelos conceituais e ontologias para representação de informações micrometeorológicas em comparação com as necessidades de sistemas relacionados (relacionados ao cenário motivador, descrito no Apêndice E).

- MARAN, Vinícius; MALDANER, Silvana; MACHADO, Guilherme Medeiros; DEGRAZIA, Gervásio; DE OLIVEIRA, José Palazzo M. **Revisão Sistemática de Modelos Conceituais e Ontologias para Representação de Dados Micrometeorológicos**. In: IX Workshop Brasileiro de Micrometeorologia, 2015, Santa Maria. DOI: 10.13140/RG.2.1.3944.1042.

No sétimo artigo foi apresentado um modelo de raciocínio para situações em *Ambient Assisted Living* que utiliza uma extensão da proposta previamente apresentada no terceiro artigo, com a adição de elementos de contexto e a modelagem de incerteza, derivada da tese de doutorado de Alencar Machado.

- MACHADO, Alencar; MARAN, Vinícius; AUGUSTIN, Iara; LIMA, João Carlos; KRUG WIVES, Leandro; DE OLIVEIRA, José Palazzo M. **Reasoning on Uncertainty in Smart Environments**. In Proceedings of the 18th International Conference on Enterprise Information Systems, ISBN 978-989-758-187-8, p. 240-250. DOI: 10.5220/0005866502400250. Classificação Qualis: B1. Este artigo foi escolhido para ser estendido como capítulo de livro em uma edição do livro *Lecture Notes in Business Information Processing* (Springer). O capítulo de livro foi submetido e o livro ainda não foi publicado.

No oitavo artigo foi apresentado o *framework* proposto nesta tese.

- MARAN, Vinícius; MACHADO, Alencar; AUGUSTIN, Iara; DE OLIVEIRA, José Palazzo M. **Semantic Integration Between Context-Awareness and Domain Data to Bring Personalized Queries to Legacy Relational Databases**. In Proceedings of the 18th International Conference on Enterprise Information Systems, ISBN 978-989-758-187-8, p. 238-243. DOI: 10.5220/0005911902380243. Classificação Qualis: B1.

Além dos artigos já publicados, foi submetido um artigo que está em fase de revisão, sem resposta até o momento:

- Para o periódico *Journal of Intelligent Information Systems* (ISSN: 1573-7675, Classificação Qualis: B1). O artigo intitulado **Linking Rules and Query Extension Algorithm to Reuse Relational Data in Ontology-Based Context-Awareness** apresenta a formalização das regras de ligação e do algoritmo extensor de consultas

apresentados nesta tese e a aplicação das regras no cenário motivador relacionado a MOOCs.

Outro artigo está em fase de escrita e correção. O artigo será submetido ao:

• Para o periódico *IET Software* (ISSN: 1751-8814, Classificação Qualis: B1). O artigo intitulado **Methodology based on Software Engineering to Integrate Ontology-Based Context-Awareness in Relational Queries** apresenta novas regras de ligação (não apresentadas no artigo anterior), a metodologia de aplicação apresentada nesta tese e a aplicação da metodologia nos cenários utilizados no capítulo 6 da tese.

REFERÊNCIAS BIBLIOGRÁFICAS

ADOMAVICIUS, Gediminas; TUZHILIN, Alexander. Context-aware recommender systems. In: **Recommender systems handbook**. Springer US, 2011. p. 217-253.

ALLIANCE, Open Mobile. User agent profile. **Approved version**, v. 2, p. 1-46, 2003. Disponível em: < http://technical.openmobilealliance.org/Technical/release_program/docs/UAProf/OMA-UAProf-V2_0-20030520-C.PDF>. Acesso em: 20 set. 2014.

ANDERSON, Kenneth M.; HANSEN, Frank Allan; BOUVIN, Niels Olof. Templates and queries in contextual hypermedia. In: **Proceedings of the seventeenth conference on Hypertext and hypermedia**. ACM, 2006. p. 99-110.

ANSTISS, Tim. Motivational interviewing in primary care. **Journal of Clinical Psychology in Medical Settings**, Springer, v. 16, n. 1, p. 87-93, 2009.

ARKOWITZ, Hal; MILLER, William R.; ROLLNICK, Stephen (Ed.). **Motivational interviewing in the treatment of psychological problems**. Guilford Publications, 2015.

ATTARD, Judie; SCERRI, S.; RIVIERA, I.; HANDSCHUH, S. Ontology-based situation recognition for context-aware systems. In: **Proceedings of the 9th International Conference on Semantic Systems**. ACM, 2013. p. 113-120.

BAUMGARTNER, N., RETSCITZEGGER , W. "A survey of upper ontologies for situation awareness." In Proceedings of the **4th IASTED International Conference on Knowledge Sharing and Collaborative Engineering**. 2006

BELLAVISTA, Paolo; CORRADI, A.; FANELLI, M. A survey of context data distribution for mobile ubiquitous systems. **ACM Computing Surveys (CSUR)**, v. 44, n. 4, p. 24, 2012.

BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora. The semantic web. **Scientific American**, v. 284, n. 5, p. 28-37, 2001.

BETTINI, Claudio; BRDICZKA, O.; HENRICKSEN, K.; INDULSKA, J.; NICKLAS, D., RANGANATHAN, A.; RIBONI, D. A survey of context modelling and reasoning techniques. **Pervasive and Mobile Computing**, v. 6, n. 2, p. 161-180, 2010.

BOLCHINI, C.; CURINO, C. A.; ORSI, G.; QUINTARELLI, E.; ROSSATO, R.; SCHREIBER, F. A.; TANCA, L. And what can context do for data?. **Communications of the ACM**, v. 52, n. 11, p. 136-140, 2009.

BOLCHINI, Cristiana; QUINTARELLI, Elisa; TANCA, Letizia. CARVE: Context-aware automatic view definition over relational databases. **Information Systems**, Elsevier, v. 38, n. 1, p. 45-67, 2013.

BORST, Willem Nico. **Construction of engineering ontologies for knowledge sharing and reuse**. Universiteit Twente, 1997.

BRÉZILLON, Patrick; ARAUJO, Renata M. de. Reinforcing shared context to improve collaboration. **Revue d'intelligence artificielle**, v. 19, n. 3, p. 537-556, 2005.

BROWN, Peter J.; JONES, Gareth JF. Context-aware retrieval: Exploring a new environment for information retrieval and information filtering. **Personal and Ubiquitous Computing**, v. 5, n. 4, p. 253-263, 2001.

BUENO-DELGADO, M. V.; PAVÓN-MARINO, P.; DE-GEA-GARCIA, A.; DOLON-GARCIA, A. The smart university experience: An NFC-based ubiquitous environment.

In: **Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on**. IEEE, 2012. p. 799-804.

BUTLER, Mark H. DELI: A delivery context library for CC/PP and UAProf. **Hp Laboratories Technical Report Hpl**, n. 260, 2001. Disponível em: < http://www.hpl.hp.com/techreports/2001/HPL2001-260.html >. Acesso em: 10 ago. 2015.

CHALMERS, Daniel. **Contextual mediation to support ubiquitous computing**. 2002. Tese de Doutorado. University of London.

CHEN, Harry, PERICH, F.; FININ, T.; JOSHI, A. Soupa: Standard ontology for ubiquitous and pervasive applications. In: **Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on**. IEEE, 2004. p. 258-267.

CHEN, Harry; FININ, Tim; JOSHI, Anupam. An ontology for context-aware pervasive computing environments. **The Knowledge Engineering Review**, v. 18, n. 03, p. 197-207, 2003.

COLACE, F.; MOSCATO, V.; QUINTARELLI, E.; RABOSIO, E.; TANCA, L. Context awareness in pervasive information management. In: **Data Management in Pervasive Systems**. Springer International Publishing, 2015. p. 235-256.

COLE, Steven; BOGENSCHUTZ, Michael; HUNGERFORD, Dan. Motivational interviewing and psychiatry: Use in addiction treatment, risky drinking and routine practice. **FOCUS: The Journal of Lifelong Learning in Psychiatry**, v. 9, n. 1, p. 42-54, 2011.

COOPER, Steve; SAHAMI, Mehran. Reflections on stanford's moocs. **Communications of the ACM**, v. 56, n. 2, p. 28-30, 2013.

DATE, Chris J.; DARWEN, Hugh. **A Guide To Sql Standard**. Reading: Addison-Wesley, 1997.

DBENGINES. Knowledge Base of Relational and NoSQL Database Management Systems. Website. Disponível em: <http://db-engines.com/en/ranking>. Acessado em: 03 mar. de 2015.

DE LABORDA, Cristian Pérez; CONRAD, Stefan. Relational. OWL: a data and schema representation format based on OWL. In: **Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43**. Australian Computer Society, Inc., 2005. p. 89-96.

DE NICOLA, Antonio; MISSIKOFF, Michele; NAVIGLI, Roberto. A software engineering approach to ontology building. **Information systems**, v. 34, n. 2, p. 258-275, 2009.

DENTLER, K., MUHLEISEN, H.; DENTLER, K., Large-Scale Storage and Reasoning for Semantic Data Using Swarms" Em: **Computational Intelligence Magazine, IEEE**, vol.7, no.2, pp.32,44, 2012.

DEY, Anind K.; ABOWD, Gregory D.; SALBER, Daniel. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. **Human-computer interaction**, v. 16, n. 2, p. 97-166, 2001.

DIAZ, A.; REGINA, M.; ROHRER, E. Making Ontology Relationships Explicit in a Ontology Network. Em: **International Workshop on Foundations of Data Management**, , v. 749, 2011.

DILLENBOURG, P.; FOX, A.; KIRCHNER, C.; MITCHELL, J.; WIRSING M. Massive Open Online Courses: Current state and perspectives (Dagstuhl Perspectives workshop 14112). **Dagstuhl Manifestos**, v. 4, n. 1, 2014.

EDX Website. Disponível em: <code.edx.org/>. Acessado em Fevereiro de 2015.

ENDSLEY, Mica R. Design and evaluation for situation awareness enhancement. In: **Proceedings of the Human Factors and Ergonomics Society Annual Meeting**. SAGE Publications, 1988. p. 97-101.

FEHLBERG, F.; ROLIM, C. O.; LEITHHARDT, V. R.; GEYER, C. F.; SILVA, L. C.; ROSSETO, A. G.. MultiS: A Context-Server for Pervasive Computing. **Electronic Notes in Theoretical Computer Science**, v. 292, p. 39-56, 2013.

FENG, Ling; APERS, Peter MG; JONKER, Willem. Towards context-aware data management for ambient intelligence. Em: **International Conference on Database and Expert Systems Applications**. Springer Berlin Heidelberg, 2004. p. 422-431.

FORTE, Marcos; DE SOUZA, Wanderley Lopes; DO PRADO, Antonio Francisco. Using ontologies and Web services for content adaptation in Ubiquitous Computing. **Journal of Systems and Software**, v. 81, n. 3, p. 368-381, 2008.

FORTUNA, Carolina et al. Metadata management for the web of things: a practical perspective. In: **Proceedings of the third international workshop on the web of things**. ACM, 2012. p. 4.

FOWLER, Martin. **Domain-specific languages**. Pearson Education, 2010.

FRIEDMAN-HILL, Ernest. **JESS in Action**. Greenwich, CT: Manning, 2003.

GAMA, Joao; GABER, Mohamed Medhat. **Learning from data streams**. Springer-Verlag Berlin Heidelberg, 2007.

GODIN, Robert; MISSAOUI, Rokia; ALAOUI, Hassan. Incremental concept formation algorithms based on Galois (concept) lattices. **Computational intelligence**, v. 11, n. 2, p. 246-267, 1995.

GRALLA, Preston. What Is Service-Oriented Architecture?. **The Web Services Advisor**, v. 6, 2003.

GRÜNINGER, Michael. Verification of the OWL-time ontology. Em: **The Semantic Web–ISWC 2011**. Springer Berlin Heidelberg, 2011. p. 225-240.

GU, Tao; PUNG, Hung Keng; ZHANG, Da Qing. A service-oriented middleware for building context-aware services. **Journal of Network and computer applications**, v. 28, n. 1, p. 1-18, 2005.

GUARINO, Nicola; CARRARA, Massimiliano; GIARETTA, Pierdaniele. Formalizing ontological commitment. Em: **AAAI**. 1994. p. 560-567.

GUTIÍRREZ-ROJAS, Israel; CRESPO-GARCÍA, Raquel M.; KLOOS, Carlos Delgado. Adapting an Awareness Tool for Massive Courses: the Case of ClassON. **Journal of Universal Computer Science**, v. 20, n. 1, p. 24-38, 2014.

HAHM, G. J.; YI, M. Y.; LEE, J. H.; SUH, H. W. A personalized query expansion approach for engineering document retrieval. **Advanced Engineering Informatics**, v. 28, n. 4, p. 344-359, 2014.

HENRICKSEN, Karen. **A framework for context-aware pervasive computing applications**. (Tese de doutorado). Queensland: University of Queensland, 2003.

HENRICKSEN, Karen; INDULSKA, Jadwiga. Developing context-aware pervasive computing applications: Models and approach. **Pervasive and mobile computing**, v. 2, n. 1, p. 37-64, 2006.

HENRICKSEN, Karen; INDULSKA, Jadwiga; MCFADDEN, Ted. Modelling context information with ORM. In: **On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops**. Springer Berlin Heidelberg, 2005. p. 626-635.

HERVÁS, Ramón; BRAVO, José; FONTECHA, Jesús. A Context Model based on Ontological Languages: a Proposal for Information Visualization. **J. UCS**, v. 16, n. 12, p. 1539-1555, 2010.

HORROCKS, Ian; PATEL-SCHNEIDER, Peter F. Reducing OWL entailment to description logic satisfiability. Em: **International Semantic Web Conference**. Springer Berlin Heidelberg, 2003. p. 17-29.

HOYOS, José Ramón; GARCÍA-MOLINA, Jesús; BOTÍA, Juan Antonio. MLContext: a context-modeling language for context-aware systems. **Electronic Communications of the EASST**, v. 28, 2010.

HUQ, M. R.; TUYEN, N. T. T.; LEE, Y. K.; JEONG, B. S.; LEE, S. Modeling an Ontology for Managing Contexts in Smart Meeting Space. In: **SWWS**. 2007. p. 96-102.

JACOBSON, Ivar et al. **The unified software development process**. Reading: Addison-wesley, 1999.

KALRA, Dipak; BEALE, Thomas; HEARD, Sam. The openEHR foundation. **Studies in health technology and informatics**, v. 115, p. 153-173, 2005.

KAYES, A. S. M.; HAN, Jun; COLMAN, Alan. An ontology-based approach to context-aware access control for software services. In: **Web Information Systems Engineering–WISE 2013**. Springer Berlin Heidelberg, 2013. p. 410-420.

KAZZAZ, M.; RYCHLY, M. Ontology-Based Context Modelling and Reasoning in the web service migration framework. **Acta Electrotechnica et Informatica**, Vol. 13, No. 4, 2013, 5–12, DOI: 10.15546/aeei-2013-0042.

KNAPPMEYER, M.; KIANI, S. L.; FRÀ, C.; MOLTCHANOV, B.; BAKER, N. ContextML: a light-weight context representation and context management schema. In: **Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on**. IEEE, 2010. p. 367-372.

KONGSAKUN, KANOKWAN; FUNG, C. C. Neural network modeling for an intelligent recommendation system supporting srm for universities in thailand. **WSEAS transactions on Computers**, v. 11, n. 2, p. 34-44, 2012.

LACLAVIK, Michal. RDB2Onto: Relational database data to ontology individuals mapping. In: **Proceeding of ninth international conference of informatics**. 2007.

LASSILA, Ora; MCGUINNESS, Deborah. The role of frame-based representation on the semantic web. **Linköping Electronic Articles in Computer and Information Science**, v. 6, n. 5, p. 2001, 2001.

LE GRAND, Bénédicte; AUFAURE, Marie-Aude; SOTO, Michel. Semantic and conceptual context-aware information retrieval. In: **Advanced Internet Based Systems and Applications**. Springer Berlin Heidelberg, 2009. p. 247-258.

LEE, Moo-Hun; RHO, Seungmin; CHOI, Eui-In. Ontology based user query interpretation for semantic multimedia contents retrieval. **Multimedia Tools and Applications**, v. 73, n. 2, p. 901-915, 2014.

LÓPEZ-NORES, M.; BLANCO-FERNANDEZ, Y.; PAZOS-ARIAS, J. J.; MARTIN-VICENTE, M. I. Context-Aware Recommender Systems Influenced by the Users' Health-

Related Data. In: **User Modeling and Adaptation for Daily Routines**. Springer London, 2013. p. 153-173.

MAARALA, Altti Ilari; Su, Xiang; Riekki, Jukka. Semantic data provisioning and reasoning for the Internet of Things. Em: **Internet of Things (IOT), 2014 International Conference on the** , vol., no., pp.67,72, 6-8 Oct. 2014.

MACHADO, Alencar. **Sensibilidade à situação em ambientes de vivência assistida: uma abordagem reativa, proativa e extensível**. Tese de doutorado. Universidade Federal do Rio Grande do Sul. 2015.

MACHADO, Guilherme M.; PALAZZO MOREIRA DE OLIVEIRA, Jose. Context-aware adaptive recommendation of resources for mobile users in a university campus. Em: **Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on**. IEEE, 2014. p. 427-433.

MAKRIS, Prodromos; SKOUTAS, Dimitrios N.; SKIANIS, Charalabos. A Survey on Context-Aware Mobile and Wireless Networking: On Networking and Computing Environments' Integration. **Communications Surveys & Tutorials, IEEE**, v. 15, n. 1, p. 362-386, 2013.

MARTINENGHI, Davide; TORLONE, Riccardo. Querying context-aware databases. Em: **Flexible Query Answering Systems**. Springer Berlin Heidelberg, 2009. p. 76-87.

MARTINS, Helio; SILVA, Nuno. Characterization, Comparison and Systematization of Context Ontologies. Em: **Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on**. IEEE, 2012. p. 983-988.

MOSTEFAOUI, Ghita Kouadri; PASQUIER-ROCHA, Jacques; BREZILLON, Patrick. Context-aware computing: a guide for the pervasive computing community. Em: **Pervasive Services, 2004. ICPS 2004. IEEE/ACS International Conference on**. IEEE, 2004. p. 39-48.

NOY, F. N.; MCGUINNESS, D. Ontology development 101: A guide to creating your first ontology. **Stantford Medical Informatics Technical Report**, SMI-2001-0880, 2001.

ORSI, Giorgio; TANCA, Letizia. Context modelling and context-aware querying. In: **Datalog Reloaded**. Springer Berlin Heidelberg, 2011. p. 225-244.

PERERA, C., S. Member, A. Zaslavsky, e P. Christen. Context aware computing for the internet of things: A survey. **Communications Surveys & Tutorials, IEEE**, v. 16, n. 1, p. 414-454, 2014.

PONCE-TOSTE, Y.; PÉREZ-GORT, M.; FERNANDEZ-PENA, F.; NUMMENMAA, J. Ontology-based Dynamic Building of Relational Data Views. Em: **Eureka-2013. Fourth International Workshop Proceedings**, v. 3, p. 4, 2013.

POVEDA VILLALON, Maria et al. A context ontology for mobile environments. **Proceedings of CIAO**. 2010.

PREUVENEERS, D.; VAN DEN BERGH, J.; WAGELAAR, D.; GEORGES, A.; RIGOLE, P.; DE BOSSCHERE, K. Towards an extensible context ontology for ambient intelligence. In: **Ambient intelligence**. Springer Berlin Heidelberg, 2004. p. 148-159.

RIBONI, Daniele; BETTINI, Claudio. Context provenance to enhance the dependability of ambient intelligence systems. **Personal and Ubiquitous Computing**, v. 16, n. 7, p. 799-818, 2012.

RIVA, Oriana. Contory: A middleware for the provisioning of context information on smart phones. In: **Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware**. Springer-Verlag New York, Inc., 2006. p. 219-239.

ROBIN, Alexandre; BOTTS, Michael E. Creation of Specific SensorML Process Models. **White Paper Earth System Science Center (NSSTC). University of Alabama in Huntsville (UAH)**, 2006.

ROBINSON, Ricky; HENRICKSEN, Karen; INDULSKA, Jadwiga. XCML: A runtime representation for the Context Modelling Language. In: **Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on**. IEEE, 2007. p. 20-26.

RODRÍGUEZ, N.; CUELLAR, M. P.; LILIUS, J.; CALVO-FLORES, M. D.. A survey on ontologies for human behavior recognition. **ACM Computing Surveys (CSUR)**, v. 46, n. 4, p. 43, 2014.

RUKZIO, E., Prezerakos, G. N., Cortese, G., Koutsoloukas, E., & Kapellaki, S. Context for Simplicity: A Basis for Context-aware Systems Based on the 3GPP Generic User Profile. **International Journal of Computational Intelligence**, v. 1, n. 1, p. 1-12, 2004.

SAHOO, Satya S. et al. A survey of current approaches for mapping of relational databases to RDF. **W3C RDB2RDF Incubator Group Report**, 2009.

SAMWALD, Matthias et al. An RDF/OWL knowledge base for query answering and decision support in clinical pharmacogenetics. **Studies in health technology and informatics**, v. 192, p. 539, 2013.

SCHILIT, Bill N.; THEIMER, Marvin M. Disseminating active map information to mobile hosts. **Network, IEEE**, v. 8, n. 5, p. 22-32, 1994.

SHENG, Quan Z.; BENATALLAH, Boualem. ContextUML: a UML-based modeling language for model-driven development of context-aware web services. In: **Mobile Business, 2005. ICMB 2005. International Conference on**. IEEE, 2005. p. 206-212.

SIRIN, Evren et al. Pellet: A practical owl-dl reasoner. **Web Semantics: science, services and agents on the World Wide Web**, v. 5, n. 2, p. 51-53, 2007.

SPIN. **SparQL Inferencin Notation**. Disponível em: <http://spinrdf.org/>. Acessado em fev. 2016.

SPORNY, Manu et al. Json-ld 1.0-a json-based serialization for linked data. **W3C Working Draft**, 2013. Disponível em: < http://travesia.mcu.es/portalnb/jspui/bitstream/10421/7478/1/JSON-LD_1_serialization.pdf >. Acessado em nov. 2014.

STAVROPOULOS, Thanos G. et al. aWESoME: A web service middleware for ambient intelligence. **Expert Systems with Applications**, v. 40, n. 11, p. 4380-4392, 2013.

STAVROPOULOS, Thanos G. et al. System architecture for a smart university building. In: **Artificial Neural Networks–ICANN 2010**. Springer Berlin Heidelberg, 2010. p. 477-482.

STERRITT, Roy; MULVENNA, Maurice; LAWRYNOWICZ, Agnieszka. Dynamic and contextualised behavioural knowledge in autonomic communications. In: **Autonomic Communication**. Springer Berlin Heidelberg, 2005. p. 217-228.

STRANG, Thomas; LINNHOFF-POPIEN, Claudia. A context modeling survey. In: **Workshop Proceedings**. 2004.

STRANG, Thomas; LINNHOFF-POPIEN, Claudia; FRANK, Korbinian. CoOL: A context ontology language to enable contextual interoperability. In: **Distributed applications and interoperable systems**. Springer Berlin Heidelberg, 2003. p. 236-247.

STRASSNER, John; O'SULLIVAN, Declan. Knowledge management for context-aware, policy-based ubiquitous computing systems. In: **Proceedings of the 6th international workshop on Managing ubiquitous communications and services**. ACM, 2009. p. 67-76.

SU, Xiang et al. Adding semantics to internet of things. **Concurrency and Computation: Practice and Experience**, v. 27, n. 8, p. 1844-1860, 2015.

SUÁREZ-FIGUEROA, Mari Carmen, et al. Essentials in ontology engineering: Methodologies, languages, and tools. 2011. Disponível em: < http://oa.upm.es/9739/1/Essentials_In_Ontology.pdf >. Acessado em jan. 2013.

SUÁREZ-FIGUEROA, Mari Carmen. **NeOn Methodology for building ontology networks: specification, scheduling and reuse**. 2010. Tese de Doutorado. Informatica.

TANCA, Letizia et al. Problems and opportunities in context-based personalization. **Proceedings of the VLDB Endowment**, v. 4, n. 11, p. 1-4, 2011.

TING, S. L. et al. RACER: Rule-Associated CasE-based Reasoning for supporting General Practitioners in prescription making. **Expert Systems with Applications**, v. 37, n. 12, p. 8079-8089, 2010.

TSARKOV, Dmitry; HORROCKS, Ian. FaCT++ description logic reasoner: System description. In: **Automated reasoning**. Springer Berlin Heidelberg, 2006. p. 292-297.

VIANA, Windson et al. Towards the semantic and context-aware management of mobile multimedia. **Multimedia Tools and Applications**, v. 53, n. 2, p. 391-429, 2011.

VYSNIAUSKAS, Ernestas; NEMURAITE, Lina. Transforming ontology representation from OWL to relational database. **Information technology and control**, v. 35, n. 3, p. 333-343, 2006.

WORLD WIDE WEB CORSOTIUM. **SWRL Overview**. Website. Disponível em: <https://www.w3.org/Submission/SWRL/>. 2004. Acessado em: Fevereiro de 2015.

WORLD WIDE WEB CORSOTIUM. **OWL 2 Web Ontology Language Document Overview** (Second Edition). Website. Disponível em: <http://www.w3.org/TR/owl2-overview/>. Acessado em: Fevereiro de 2015.

WANG, H., Zhang, R., & Wang, Z. (2012). JenaPro: A Distributed File Storage Engine for Jena. 2012 **Fifth International Joint Conference on Computational Sciences and Optimization**, 610–613.

WANG, Xiao Hang et al. Ontology based context modeling and reasoning using OWL. In: **Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on**. Ieee, 2004. p. 18-22.

WEISER, Mark. The computer for the 21st century. **Scientific american**, v. 265, n. 3, p. 94-104, 1991. Disponível em: http://doi.acm.org/10.1145/329124.329126.

YE, Juan; DOBSON, Simon; MCKEEVER, Susan. Situation identification techniques in pervasive computing: A review. **Pervasive and mobile computing**, v. 8, n. 1, p. 36-66, 2012.

YILMAZ, Özgün; ERDUR, Rıza Cenk. iConAwa–An intelligent context-aware system. **Expert Systems with Applications**, v. 39, n. 3, p. 2907-2918, 2012.

ZHANG, Xutang et al. Ontology-based semantic retrieval for engineering domain knowledge. **Neurocomputing**, v. 116, p. 382-391, 2013.

ZIMMERMANN, Andreas; LORENZ, Andreas; OPPERMANN, Reinhard. An operational definition of context. Em: **Modeling and using context**. Springer Berlin Heidelberg, 2007. p. 558-571.

APÊNDICE A – MODELO LÓGICO DA PLATAFORMA OPENEDX UTILIZADA NOS TESTES

assessment_rubric (id, content_hash)
assessment_assessment (id, submission_uuid, rubric_id, scored_at, scorer_id, score_type, feedback)
rubric_id REFERENCES assessment_rubric
assessment_assessmentfeedback (id, submission_uuid, feedback_text)
assessment_assessmentfeedback_assessments (id, assessmentfeedback_id, assessment_id,
assessmentfeedback_id REFERENCES assessment_assessmentfeedback
assessment_id REFERENCES assessment_assessment
assessment_assessmentfeedbackoption (id, text)
assessment_assessmentfeedback_options (id, assessmentfeedback_id, assessmentfeedbackoption_id)
assessmentfeedbackoption_id REFERENCES assessment_assessmentfeedbackoption
assessmentfeedback_id REFERENCES assessment_assessmentfeedback
assessment_criterion (id, rubric_id, name, order_num, prompt)
rubric_id REFERENCES assessment_rubric
assessment_criterionoption (id, criterion_id, order_num, points, name, explanation)
criterion_id REFERENCES assessment_criterion
assessment_assessmentpart (id, assessment_id, option_id, feedback)
assessment_id REFERENCES assessment_assessment
option_id REFERENCES assessment_criterionoption
assessment_peerworkflow (id, student_id, item_id, course_id, submission_uuid, created_at, completed_at, grading_completed_at)
assessment_peerworkflowitem (id, scorer_id, author_id, submission_uuid, started_at, assessment_id, scored)
assessment_id REFERENCES assessment_assessment)
author_id REFERENCES assessment_peerworkflow
scorer_id REFERENCES assessment_peerworkflow
assessment_studenttrainingworkflow (id, submission_uuid, student_id, item_id, course_id)
assessment_trainingexample (id, raw_answer, rubric_id, content_hash)
rubric_id REFERENCES assessment_rubric
assessment_studenttrainingworkflowitem (id, workflow_id, order_num, started_at, completed_at, training_example_id)
training_example_id REFERENCES assessment_trainingexample
workflow_id REFERENCES assessment_studenttrainingworkflow
assessment_trainingexample_options_selected (id, trainingexample_id, criterionoption_id)
criterionoption_id REFERENCES assessment_criterionoption
trainingexample_id REFERENCES assessment_trainingexample
auth_group (id, name)
django_content_type (id, name, app_label, model)
auth_permission (id, name, content_type_id, codename)
content_type_id REFERENCES django_content_type
auth_group_permissions (id, group_id, permission_id)
group_id REFERENCES auth_group
permission_id REFERENCES auth_permission
auth_user (id, username, first_name, last_name, email, password, is_staff, is_active, is_superuser, last_login, date_joined)
auth_registration (id, user_id, activation_key)
user_id REFERENCES auth_user
auth_user_groups (id, user_id, group_id)
group_id REFERENCES auth_group
user_id REFERENCES auth_user
auth_user_user_permissions (id, user_id, permission_id)
permission_id REFERENCES auth_permission
user_id REFERENCES auth_user
auth_userprofile (id, user_id, name, language, location, meta, courseware, gender, mailing_address, year_of_birth, level_of_education,
goals, allow_certificate, country, city)
user_id REFERENCES auth_user
bulk_email_courseauthorization (id, course_id, email_enabled)
bulk_email_courseemail (id, sender_id, slug, subject, html_message, created, modified, course_id, to_option, text_message)
sender_id REFERENCES auth_user
bulk_email_courseemailtemplate (id, html_template, plain_template)
bulk_email_optout (id, course_id, user_id)
user_id REFERENCES auth_user
celery_taskmeta (id, task_id, status, result, date_done, traceback, hidden, meta)
celery_tasksetmeta (id, taskset_id, result, date_done, hidden)
certificates_certificatewhitelist (id, user_id, course_id, whitelist)
user_id REFERENCES auth_user
certificates_generatedcertificate (id, user_id, download_url, grade, course_id, key, distinction, status, verify_uuid, download_uuid, name,
created_date, modified_date, error_reason, mode)
user_id REFERENCES auth_user
circuit_servercircuit (id, name, schematic)
course_creators_coursecreator (id, user_id, state_changed, state, note)
user_id REFERENCES auth_user
course_groups_courseusergroup (id, name, course_id, group_type)
course_groups_courseusergroup_users (id, courseusergroup_id, user_id)

courseusergroup_id REFERENCES course_groups_courseusergroup
user_id REFERENCES auth_user
course_modes_coursemode (id, course_id, mode_slug, mode_display_name, min_price, suggested_prices, currency, expiration_date, expiration_datetime)
courseware_offlinecomputedgrade (id, user_id, course_id, created, updated, gradeset)
user_id REFERENCES auth_user
courseware_offlinecomputedgradelog (id, course_id, created, seconds, nstudents)
courseware_studentmodule (id, module_type, module_id, student_id, state, grade, created, modified, max_grade, done, course_id)
student_id REFERENCES auth_user
courseware_studentmodulehistory (id, student_module_id, version, created, state, grade, max_grade)
student_module_id REFERENCES courseware_studentmodule
courseware_xmodulestudentinfofield (id, field_name, value, student_id, created, modified)
student_id REFERENCES auth_user
courseware_xmodulestudentprefsfield (id, field_name, module_type, value, student_id, created, modified)
student_id REFERENCES auth_user
courseware_xmoduleuserstatesummaryfield (id, field_name, usage_id, value, created, modified)
dark_lang_darklangconfig (id, change_date, changed_by_id, enabled, released_languages)
changed_by_id REFERENCES auth_user
django_admin_log (id, action_time, user_id, content_type_id, object_id, object_repr, action_flag, change_message)
content_type_id REFERENCES django_content_type
user_id REFERENCES auth_user
django_comment_client_permission (name)
django_comment_client_role (id, name, course_id)
django_comment_client_permission_roles (id, permission_id, role_id)
permission_id REFERENCES django_comment_client_permission
role_id REFERENCES django_comment_client_role
django_comment_client_role_users (id, role_id, user_id)
role_id REFERENCES django_comment_client_role
user_id REFERENCES auth_user
django_openid_auth_association (id, server_url, handle, secret, issued, lifetime, assoc_type)
django_openid_auth_nonce (id, server_url, timestamp, salt)
django_openid_auth_useropenid (id, user_id, claimed_id, display_id)
user_id REFERENCES auth_user
django_session (session_key, session_data, expire_date)
django_site (id, domain, name)
djcelery_crontabschedule (id, minute, hour, day_of_week, day_of_month, month_of_year)
djcelery_intervalschedule (id, every, period)
djcelery_periodictask (id, name, task, interval_id, crontab_id, args, kwargs, queue, exchange, routing_key, expires, enabled, last_run_at, total_run_count, date_changed, description)
crontab_id REFERENCES djcelery_crontabschedule
interval_id REFERENCES djcelery_intervalschedule
djcelery_periodictasks (ident, last_update)
djcelery_workerstate (id, hostname, last_heartbeat)
djcelery_taskstate (id, state, task_id, name, tstamp, args, kwargs, eta, expires, result, traceback, runtime, retries, worker_id, hidden)
worker_id REFERENCES djcelery_workerstate
embargo_embargoedcourse (id, course_id, embargoed)
embargo_embargoedstate (id, change_date, changed_by_id, enabled, embargoed_countries)
changed_by_id REFERENCES auth_user
embargo_ipfilter (id, change_date, changed_by_id, enabled, whitelist, blacklist)
changed_by_id REFERENCES auth_user
experiments_anonymouspost (id, user, commentid)
experiments_strategyrandomization (id, strategyType, percents, planoutScript, planoutJson, customDesign, fatorial, periodos, periodoRel_id, clusterGroups)
periodoRel_id REFERENCES experiments_experimentdefinition
experiments_experimentdefinition (id, descricao, course, userTeacher_id, strategy_id)
strategy_id REFERENCES experiments_strategyrandomization
userTeacher_id REFERENCES auth_user
experiments_opcoesexperiment (id, experimento_id, sectionExp, sectionExp_url, version)
experimento_id REFERENCES experiments_experimentdefinition
experiments_groupscluster (id, grupos, experiment_id, versao_id)
experiment_id REFERENCES experiments_experimentdefinition
versao_id REFERENCES experiments_opcoesexperiment
experiments_historicoquestoes (id, campo, valor, usuario_id)
usuario_id REFERENCES auth_user
experiments_userchoiceexperiment (id, userStudent_id, versionExp_id, bloco, experimento_id)
experimento_id REFERENCES experiments_experimentdefinition
userStudent_id REFERENCES auth_user
versionExp_id REFERENCES experiments_opcoesexperiment
external_auth_externalauthmap (id, external_id, external_domain, external_credentials, external_email, external_name, user_id, internal_password, dtcreated, dtsignup)
user_id REFERENCES auth_user
foldit_puzzlecomplete (id, user_id, unique_user_id, puzzle_id, puzzle_set, puzzle_subset, created)
user_id REFERENCES auth_user
foldit_score (id, user_id, unique_user_id, puzzle_id, best_score, current_score, score_version, created)
user_id REFERENCES auth_user
instructor_task_instructortask (id, task_type, course_id, task_key, task_input, task_id, task_state, task_output, requester_id, created, updated,

subtasks)
    requester_id REFERENCES auth_user
licenses_coursesoftware (id, name, full_name, url, course_id)
licenses_userlicense (id, software_id, user_id, serial)
    software_id REFERENCES licenses_coursesoftware
    user_id REFERENCES auth_user
linkedin_linkedin (user_id, has_linkedin_account, emailed_courses)
    user_id REFERENCES auth_user
notes_note (id, user_id, course_id, uri, text, quote, range_start, range_start_offset, range_end, range_end_offset, tags, created, updated)
    user_id REFERENCES auth_user
wiki_articlerevision (id, revision_number, user_message, automatic_log, ip_address, user_id, modified, created, previous_revision_id,
deleted, locked, article_id, content, title)
    article_id REFERENCES wiki_article
    previous_revision_id REFERENCES wiki_articlerevision
    user_id REFERENCES auth_user
wiki_article (id, current_revision_id, created, modified, owner_id, group_id, group_read, group_write, other_read, other_write)
    current_revision_id REFERENCES wiki_articlerevision
    group_id REFERENCES auth_group
    owner_id REFERENCES auth_user
wiki_articleplugin (id, article_id, deleted, created)
    article_id REFERENCES wiki_article
notify_notificationtype (key, label, content_type_id)
    content_type_id REFERENCES django_content_type
notify_settings (id, user_id, interval)
    user_id REFERENCES auth_user
notify_subscription (id, settings_id, notification_type_id, object_id, send_emails)
    notification_type_id REFERENCES notify_notificationtype
    settings_id REFERENCES notify_settings
notifications_articlesubscription (subscription_ptr_id, articleplugin_ptr_id)
    articleplugin_ptr_id REFERENCES wiki_articleplugin
    subscription_ptr_id REFERENCES notify_subscription
notify_notification (id, subscription_id, message, url, is_viewed, is_emailed, created)
    subscription_id REFERENCES notify_subscription
psychometrics_psychometricdata (id, studentmodule_id, done, attempts, checktimes)
reverification_midcoursereverificationwindow (id, course_id, start_date, end_date)
student_courseenrollment (id, user_id, course_id, created, is_active, mode)
    user_id REFERENCES auth_user
shoppingcart_order (id, user_id, currency, status, purchase_time, bill_to_first, bill_to_last, bill_to_street1, bill_to_street2, bill_to_city,
bill_to_state, bill_to_postalcode, bill_to_country, bill_to_ccnum, bill_to_cardtype, processor_reply_dump, refunded_time)
    user_id REFERENCES auth_user
shoppingcart_orderitem (id, order_id, user_id, status, qty, unit_cost, line_desc, currency, fulfilled_time, report_comments,
refund_requested_time, service_fee)
    order_id REFERENCES shoppingcart_order
    user_id REFERENCES auth_user
shoppingcart_certificateitem (orderitem_ptr_id, course_id, course_enrollment_id, mode)
    course_enrollment_id REFERENCES student_courseenrollment
    orderitem_ptr_id REFERENCES shoppingcart_orderitem
shoppingcart_paidcourseregistration (orderitem_ptr_id, course_id, mode)
    orderitem_ptr_id REFERENCES shoppingcart_orderitem
shoppingcart_paidcourseregistrationannotation (id, course_id, annotation)
south_migrationhistory (id, app_name, migration, applied)
splash_splashconfig (id, change_date, changed_by_id, enabled, cookie_name, cookie_allowed_values, unaffected_usernames, redirect_url,
unaffected_url_paths)
    changed_by_id REFERENCES auth_user
student_anonymoususerid (id, user_id, anonymous_user_id, course_id)
    user_id REFERENCES auth_user
student_courseaccessrole (id, user_id, org, course_id, role)
    user_id REFERENCES auth_user
student_courseenrollmentallowed (id, email, course_id, created, auto_enroll)
student_loginfailures (id, user_id, failure_count, lockout_until)
    user_id REFERENCES auth_user
student_passwordhistory (id, user_id, password, time_set)
    user_id REFERENCES auth_user
student_pendingemailchange (id, user_id, new_email, activation_key)
    user_id REFERENCES auth_user
student_pendingnamechange (id, user_id, new_name, rationale)
    user_id REFERENCES auth_user
student_userstanding (id, user_id, account_status, changed_by_id, standing_last_changed_at)
    changed_by_id REFERENCES auth_user
    user_id REFERENCES auth_user
student_usertestgroup (id, name, description)
student_usertestgroup_users (id, usertestgroup_id, user_id)
    usertestgroup_id REFERENCES student_usertestgroup
    user_id REFERENCES auth_user
submissions_studentitem (id, student_id, course_id, item_id, item_type)
submissions_submission (id, uuid, student_item_id, attempt_number, submitted_at, created_at, raw_answer)

student_item_id REFERENCES submissions_studentitem
submissions_score (id, student_item_id, submission_id, points_earned, points_possible, created_at, reset)
student_item_id REFERENCES submissions_studentitem
submission_id REFERENCES submissions_submission
submissions_scoresummary (id, student_item_id, highest_id, latest_id)
highest_id REFERENCES submissions_score
latest_id REFERENCES submissions_score
student_item_id REFERENCES submissions_studentitem
track_trackinglog (id, dtcreated, username, ip, event_source, event_type, event, agent, page, time, host)
user_api_usercoursetag (id, user_id, key, course_id, value)
user_id REFERENCES auth_user
user_api_userpreference (id, user_id, key, value)
user_id REFERENCES auth_user
verify_student_softwaresecurephotoverification (id, status, status_changed, user_id, name, face_image_url, photo_id_image_url, receipt_id, created_at, updated_at, submitted_at, reviewing_user_id, reviewing_service, error_msg, error_code, photo_id_key, window_id, display)
reviewing_user_id REFERENCES auth_user
user_id REFERENCES auth_user
window_id REFERENCES reverification_midcoursereverificationwindow
waffle_flag (id, name, everyone, percent, superusers, staff, authenticated, rollout, note, testing, created, modified, languages)
waffle_flag_groups (id, flag_id, group_id)
flag_id REFERENCES waffle_flag
group_id REFERENCES auth_group
waffle_flag_users (id, flag_id, user_id)
flag_id REFERENCES waffle_flag
user_id REFERENCES auth_user
waffle_sample (id, name, percent, note, created, modified)
waffle_switch (id, name, active, note, created, modified)
wiki_articleforobject (id, article_id, content_type_id, object_id, is_mptt)
article_id REFERENCES wiki_article
content_type_id REFERENCES django_content_type
wiki_articlesubscription (subscription_ptr_id, articleplugin_ptr_id)
articleplugin_ptr_id REFERENCES wiki_articleplugin
subscription_ptr_id REFERENCES notify_subscription
wiki_attachmentrevision (id, revision_number, user_message, automatic_log, ip_address, user_id, modified, created, previous_revision_id, deleted, locked, attachment_id, file, description)
attachment_id REFERENCES wiki_attachment
previous_revision_id REFERENCES wiki_attachmentrevision
user_id REFERENCES auth_user
wiki_reusableplugin (articleplugin_ptr_id)
articleplugin_ptr_id REFERENCES wiki_articleplugin
wiki_attachment (reusableplugin_ptr_id, current_revision_id, original_filename)
current_revision_id REFERENCES wiki_attachmentrevision
reusableplugin_ptr_id REFERENCES wiki_reusableplugin
wiki_revisionpluginrevision (id, revision_number, user_message, automatic_log, ip_address, user_id, modified, created, previous_revision_id, deleted, locked, plugin_id)
plugin_id REFERENCES wiki_revisionplugin
previous_revision_id REFERENCES wiki_revisionpluginrevision
user_id REFERENCES auth_user
wiki_revisionplugin (articleplugin_ptr_id, current_revision_id)
articleplugin_ptr_id REFERENCES wiki_articleplugin
current_revision_id REFERENCES wiki_revisionpluginrevision
wiki_image (revisionplugin_ptr_id)
revisionplugin_ptr_id REFERENCES wiki_revisionplugin
wiki_imagerevision (revisionpluginrevision_ptr_id, image, width, height)
revisionpluginrevision_ptr_id REFERENCES wiki_revisionpluginrevision
wiki_reusableplugin_articles (id, reusableplugin_id, article_id)
article_id REFERENCES wiki_article
reusableplugin_id REFERENCES wiki_reusableplugin
wiki_simpleplugin (articleplugin_ptr_id, article_revision_id)
articleplugin_ptr_id REFERENCES wiki_articleplugin
article_revision_id REFERENCES wiki_articlerevision
wiki_urlpath (id, slug, site_id, parent_id, lft, rght, tree_id, level, article_id)
article_id REFERENCES wiki_article
parent_id REFERENCES wiki_urlpath
site_id REFERENCES django_site
workflow_assessmentworkflow (id, created, modified, status, status_changed, submission_uuid, uuid, course_id, item_id)
workflow_assessmentworkflowstep (id, workflow_id, name, submitter_completed_at, assessment_completed_at, order_num)
workflow_id REFERENCES workflow_assessmentworkflow

APÊNDICE B – CONFIGURAÇÕES DO SERVIDOR E DOS SOFTWARES UTILIZADOS NOS TESTES

Servidor Utilizado nos Testes:

Servidor DELL, modelo R630. 16gb de RAM e 2 processadores Intel Xeon E5-2600.

Softwares Utilizados nos Testes:

Software de Virtualização XEN Server. http://xenserver.org

Linux Ubuntu v12.04.

Plataforma OpenEDX. Versão Dev Stack.
https://openedx.atlassian.net/wiki/display/OpenOPS/Running+Devstack

Java JRE. Versão 8.

MySQL. Versão 5.7.14

MySQL JDBC Connector. 6.01

Protégé Versão 5.01

Pellet Reasoner.

OWL API. Versão 4.02

APÊNDICE C – VOCABULÁRIO IMPORTADO DA ONTOLOGIA

Ability	Cgnitive Ability	Physical Ability	Activity	Characteristics
Contact	Device	Mobile	Motionless	DeviceFeature
App	Connection	GPS	Screen	Sound
Education	Element	Annotation	Classification	Educational
General	Lifecycle	Metametadata	Rights	Techniacl
ElementComponent	Contribute	Identifier	Orcomposite	Requirement
Taxon	TaxonPath	Entity	RuleGroup	Environment
EnvironmentType	Expertise	ComputerExpertise	Interest	LearningObject
Living Conditions	Location	LOM	Organization	Intitute
Program	Researh Group	School	University	University Department
Person	Academic	Administrative	PossedThing	Living Thing
Non Living Thing	Preference	Profession	Publication	Article
Conference Paper	Journal Article	Book	Manual	Role
Professor	Student	Schedule	Subject of Interest	System
Technology	User	Work	Research Work	Teaching Course
Graduate Level Course	Employee	Clerical Staff Worker	Associate Professor	

APÊNDICE D – CONSULTAS ESTENDIDAS APÓS APLICAÇÃO DO MODELO

CONSULTA 1 no CONTEXTO DE INTERESSE A:

```
SELECT t1.idmodule AS lev1, t2.idmodule AS lev2, t3.idmodule AS lev3, t4.idmodule
AS lev4, def.definition AS lev4definition FROM module_child AS t1 LEFT JOIN
module_child AS t2 ON t2.idmodule = t1.idchild LEFT JOIN module_child AS t3 ON
t3.idmodule = t2.idchild LEFT JOIN module_child AS t4 ON t4.idmodule = t3.idchild
JOIN module ON t4.idmodule = module.id JOIN module_definition AS def ON
module.definition=def.id WHERE t1.idmodule = '57bb8878e4efae083b63c157' AND
(module_definition.definition->"$.block_type" <> 'video' OR
module_definition.definition->"$.block_info.duration" < 600 OR
module_definition.definition->"$.block_type" <> 'video')
```

CONSULTA 2 no CONTEXTO DE INTERESSE A:

```
SELECT course.id, t1.idmodule AS lev1, t2.idmodule AS lev2,
module_definition.definition FROM auth_user JOIN student_courseenrollment ON
(auth_user.id = student_courseenrollment.user_id) JOIN course ON
(student_courseenrollment.course_id = course.id) JOIN module_child AS t1 ON
(t1.idmodule = course.module_id) LEFT JOIN module_child AS t2 ON (t2.idmodule =
t1.idchild) JOIN module ON (t2.idmodule = module.id) JOIN module_definition ON
module.definition = module_definition.id WHERE auth_user.id = 5 AND
course.id='course-v1:UnivTest+EM101+2016_1' AND (module_definition.definition-
>"$.block_type" <> 'video' OR module_definition.definition-
>"$.block_info.duration" < 600 OR module_definition.definition->"$.block_type" <>
'video')
```

CONSULTA 1 no CONTEXTO DE INTERESSE B:

```
SELECT t1.idmodule AS lev1, t2.idmodule AS lev2, t3.idmodule AS lev3, t4.idmodule
AS lev4, module_definition.definition AS lev4definition  FROM module_child AS t1
LEFT JOIN module_child AS t2 ON t2.idmodule = t1.idchild LEFT JOIN module_child AS
t3 ON t3.idmodule = t2.idchild LEFT JOIN module_child AS t4 ON t4.idmodule =
t3.idchild JOIN module ON t4.idmodule = module.id JOIN module_definition ON
module.definition=module_definition.id WHERE t1.idmodule =
'57bb8878e4efae083b63c157' AND (module_definition.definition->"$.block_type" =
'overview' OR module_definition.definition->"$.block_type" = 'vertical' OR
module_definition.definition->"$.block_type" = 'course' OR module.id = '
3a48cceb756f4bc0b9f0dd71ce1eaa19' OR module.id = '
3e9a630776124e99bc99981a4b059c62' OR module.id = '
415a6ac9bfc4473b966373bc93dc1188' OR module.id = '
4d3dc080d95847a4a2ce08f6787ae41e' OR module.id = '
5a789820e1104d1faae15ce505645f4e'OR module.id = '
5edf5b08dbf74eb3ae6b7d5a610f2233' OR module.id = '
61f8dc1e35d74b3c9ad616bc71ad7ca8' OR module.id = '
64c7fedd61d34966b22ef8274d680041' OR module.id = '
67df3f7dc2564f359d74c568f8af5801' OR module.id = '
6cdda1f46cc44af3a2a6a538682e93b0' OR module.id = '
7463bad035114ad89998721a8219059c' OR module.id = '
846d7c85e3654c4691910c8831fcc7f5' OR module.id = '
846d7c85e3654c4691910c8831fcc7f5' OR module.id = '
846d7c85e3654c4691910c8831fcc7f5'OR module_definition.definition->"$.block_type" =
'overview' OR module_definition.definition->"$.block_type" = 'vertical' OR
module_definition.definition->"$.block_type" = 'course' OR module.id = '
8908bc18fb424a0d8c3acda022d141ea'OR module.id = '
920fcc05a4b34b909b7c8ea1207320ec'OR module.id = '
a03572cb9b2c433095934b3a806558a0'OR module.id = '
a2a04bd62fef455a960d8bea83424f60'OR module.id = '
b07ff10b96e146c5ab9a14092a769c81' OR module.id = '
```

```
d9f94df02284465c906ab4998fd22132' OR module.id = '
e5017a909b0e4ec2945a6ff7814943c8')
```

CONSULTA 2 no CONTEXTO DE INTERESSE B:

```
SELECT course.id, t1.idmodule AS lev1, t2.idmodule AS lev2,
module_definition.definition FROM auth_user JOIN student_courseenrollment ON
(auth_user.id = student_courseenrollment.user_id) JOIN course ON
(student_courseenrollment.course_id = course.id) JOIN module_child AS t1 ON
(t1.idmodule = course.module_id) LEFT JOIN module_child AS t2 ON (t2.idmodule =
t1.idchild) JOIN module ON (t2.idmodule = module.id) JOIN module_definition ON
module.definition = module_definition.id WHERE auth_user.id = 5 AND
course.id='course-v1:asd+asd+asd' AND (module_definition.definition-
>"$.block_type" = 'overview' OR module_definition.definition->"$.block_type" =
'vertical' OR module_definition.definition->"$.block_type" = 'course' OR module.id
= ' 3a48cceb756f4bc0b9f0dd71ce1eaa19' OR module.id = '
3e9a630776124e99bc99981a4b059c62' OR module.id = '
415a6ac9bfc4473b966373bc93dc1188' OR module.id = '
4d3dc080d95847a4a2ce08f6787ae41e' OR module.id = '
5a789820e1104d1faae15ce505645f4e'OR module.id = '
5edf5b08dbf74eb3ae6b7d5a610f2233' OR module.id = '
61f8dc1e35d74b3c9ad616bc71ad7ca8' OR module.id = '
64c7fedd61d34966b22ef8274d680041' OR module.id = '
67df3f7dc2564f359d74c568f8af5801' OR module.id = '
6cdda1f46cc44af3a2a6a538682e93b0' OR module.id = '
7463bad035114ad89998721a8219059c' OR module.id = '
846d7c85e3654c4691910c8831fcc7f5' OR module.id = '
846d7c85e3654c4691910c8831fcc7f5' OR module.id = '
846d7c85e3654c4691910c8831fcc7f5'OR module_definition.definition->"$.block_type" =
'overview' OR module_definition.definition->"$.block_type" = 'vertical' OR
module_definition.definition->"$.block_type" = 'course' OR module.id = '
8908bc18fb424a0d8c3acda022d141ea'OR module.id = '
920fcc05a4b34b909b7c8ea1207320ec'OR module.id = '
a03572cb9b2c433095934b3a806558a0'OR module.id = '
a2a04bd62fef455a960d8bea83424f60'OR module.id = '
b07ff10b96e146c5ab9a14092a769c81'OR module.id = '
d9f94df02284465c906ab4998fd22132'OR module.id = '
e5017a909b0e4ec2945a6ff7814943c8')
```

CONSULTA 1 no CONTEXTO DE INTERESSE C:

```
SELECT t1.idmodule AS lev1, t2.idmodule AS lev2, t3.idmodule AS lev3, t4.idmodule
AS lev4, def.definition AS lev4definition FROM module_child AS t1 LEFT JOIN
module_child AS t2 ON t2.idmodule = t1.idchild LEFT JOIN module_child AS t3 ON
t3.idmodule = t2.idchild LEFT JOIN module_child AS t4 ON t4.idmodule = t3.idchild
JOIN module ON t4.idmodule = module.id JOIN module_definition AS def ON
module.definition=def.id WHERE t1.idmodule = '57bb8878e4efae083b63c157' AND
(module.id = '15a6702475864fdaadc00a9814e7e094' OR module.id =
'374331404a324c3b8e93666298dbaf1e' OR module.id =
'bfa49b8080724351842042 63f97b273f' OR module.id =
'f53578e5932b431886e54371c3051c3c' OR module.id =
'f1cad38d8a264fdea3daab2b5abb993e' OR module_definition.definition->"$.block_type"
= 'overview' OR module_definition.definition->"$.block_type" = 'discussion' OR
module_definition.definition->"$.block_type" = 'course')
```

CONSULTA 2 no CONTEXTO DE INTERESSE C:

```
SELECT course.id, t1.idmodule AS lev1, t2.idmodule AS lev2,
module_definition.definition FROM auth_user JOIN student_courseenrollment ON
(auth_user.id = student_courseenrollment.user_id) JOIN course ON
(student_courseenrollment.course_id = course.id)
```

```
JOIN module_child AS t1 ON (t1.idmodule = course.module_id) LEFT JOIN module_child
AS t2 ON (t2.idmodule = t1.idchild) JOIN module ON (t2.idmodule = module.id)
JOIN module_definition ON module.definition = module_definition.id WHERE
auth_user.id = 5 AND course.id='course-v1:UnivTest+EM101+2016_1'
AND (module.id = '15a6702475864fdaadc00a9814e7e094' OR module.id =
'374331404a324c3b8e93666298dbaf1e' OR module.id =
'bfa49b80807243518420426 3f97b273f' OR module.id =
'f53578e5932b431886e54371c3051c3c' OR module.id =
'f1cad38d8a264fdea3daab2b5abb993e' OR module_definition.definition->"$.block_type"
= 'overview' OR module_definition.definition->"$.block_type" = 'discussion' OR
module_definition.definition->"$.block_type" = 'course')
```

APÊNDICE E – CENÁRIO MOTIVADOR: DISTRIBUIÇÃO CONTEXTUALIZADA DE INFORMAÇÕES METEOROLÓGICAS

Engenharia de Conhecimento é uma área da computação que trata de processos, metodologias, ferramentas e tecnologias de suporte ao gerenciamento de conhecimento em sistemas computacionais. Ontologias são artefatos utilizados para suportar a Engenharia do Conhecimento, e podem ser definidas como uma especificação formal e explícita de uma conceituação compartilhada (Studer et al., 1998).

Especificações formais são compostas por axiomas que representam formalmente o conhecimento capturado. Para que isto ocorra, elas devem ser feitas utilizando uma linguagem formal, passível de processamento por máquinas. Estas linguagens por sua vez possuem um determinado nível de expressividade. Ontologias devem ser aceitas e compartilhadas por pelo menos, um grupo de pessoas (Studer et al., 1998).

Ontologias têm diversas aplicações em sistemas computacionais. Uma das mais empregadas é o uso para descrição de domínios em um alto nível de abstração. Esta metodologia permite que uma descrição formal, de um domínio, utilizada por uma determinada aplicação, possa ser reaproveitada não só por pessoas, mas também por agentes computacionais inteligentes (Staab; Studer, 2013).

No caso em estudo, estações micrometeorológicas coletam dados provenientes da atmosfera e permitem que estas medições auxiliem no entendimento de fenômenos climáticos (Serafini, 2008). Os dados originados de estações podem ser utilizados em diversos tipos de aplicações, como por exemplo: (i) Arquiteturas de gerenciamento de processos de agricultura de precisão (Morgenstern et al., 2015) (Ma et al., 2014), (ii) Arquiteturas de distribuição de dados meteorológicos (Zink et al., 2010), e (iii) Serviços distribuídos de gerenciamento de dados meteorológicos (Aldaz et al., 2008).

Devido a necessidade da utilização de informações provenientes de estações micrometeorológicas por diversos tipos de sistemas, surge a necessidade da existência de representações semânticas destas informações, para que as mesmas possam ser utilizadas por sistemas ou agentes inteligentes.

A representação destas informações pode ser feita através da criação de ontologias que descrevem os sensores e os tipos de dados coletados por estes sensores. Estas informações são utilizadas pelos sistemas e agentes inteligentes em diferentes níveis de abstração.

Por exemplo, é interessante para arquiteturas de distribuição de dados meteorológicos que as informações sejam descritas em baixos níveis de abstração, unindo dados brutos às suas representações semânticas. Já para outros serviços, como por exemplo, serviços distribuídos de gerenciamento de dados meteorológicos, é interessante que as informações sejam disponibilizadas em um nível de abstração mais alto, como por exemplo A probabilidade de chover mais de 50mm diários entre os dias 10 e 12 do mês corrente.

O padrão ODM2 (Hosburgh et al., 2016) foi proposto como uma extensão do padrão ODM1. Ele apresenta um conjunto de conceitos, atributos e relações definidos para facilitar o processo de troca de informações sobre observações meteorológicas por empresas e grupos de pesquisa. Observações são definidas como *“Uma observação é um ato associado com um instante de tempo discreto ou período através do qual um número, termo, ou outro símbolo é atribuído a um fenómeno. Envolve a aplicação de um processo especificado, tal como um sensor, instrumento, o algoritmo, ou cadeia de processo. O procedimento pode ser aplicado em tempo real na aplicação, ou remotamente, no que diz respeito à localização de*

amostragem. O resultado de uma observação é uma estimativa do valor de uma propriedade de alguma característica" (Hosburgh et al., 2016).

O padrão ODM2 é representado em esquemas Entidade-Relacionamento, com a definição de tabelas, atributos e relacionamentos destas tabelas. O esquema principal é chamado de *core*, composto por um conjunto de tabelas, apresentadas na Figura 1.

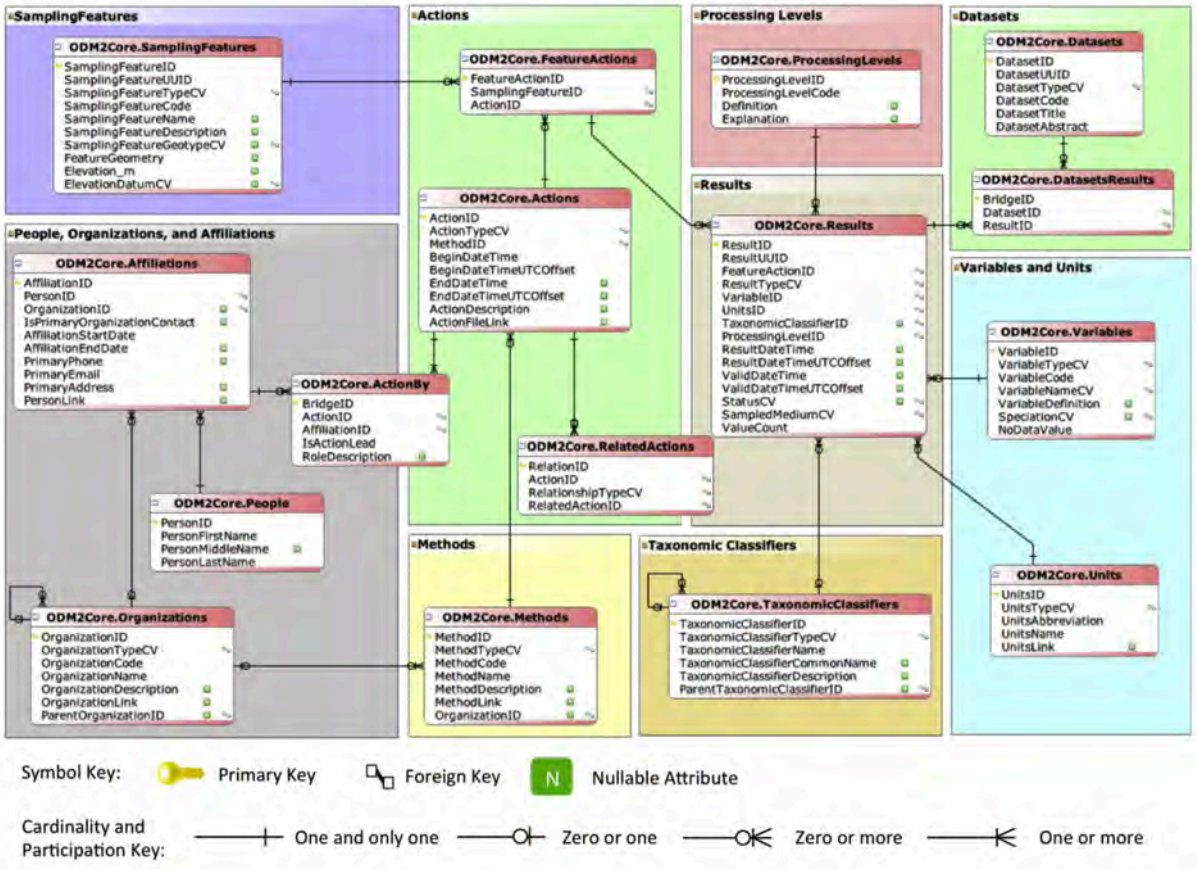


Figura 1 – Esquema ODM2 *Core*.

Uma visão geral do cenário motivador é apresentado na Figura 2. Vamos considerar a existência de uma arquitetura de gerenciamento de dados relacionados a observações micrometeorológicas. Esta arquitetura gerencia um banco de dados relacional que utiliza o padrão ODM2 para representação de observações meteorológicas. Uma implementação da arquitetura é gerenciada por um instituto meteorológico, que recebe informações meteorológicas coletadas por sensores espalhados por diversas localidades no estado do Rio Grande do Sul.

Um agricultor (a) utiliza uma aplicação móvel que se comunica com a arquitetura U-Agro (b). A arquitetura U-Agro (Morgenstern et al., 2015) é uma arquitetura de software que realiza recomendações para agricultores que utilizam os processos de agricultura de precisão em suas atividades. As recomendações são realizadas utilizando uma ontologia de contexto, modelada na linguagem OWL-DL. Um conjunto definido de regras no formato SWRL define operações de recomendação, baseada em informações de contexto.

A arquitetura de gerenciamento de observações recebe informações sobre eventos climáticos (e) captados por uma série de sensores (f) e realiza a persistência de informações

meteorológicas (g), e as distribui para outras arquiteturas de acordo com o contexto informado por estas arquiteturas. Em um determinado momento, a arquitetura U-Agro realiza uma consulta relacional (baseada no modelo do formato ODM2) (c) na arquitetura de gerenciamento de observações. Além disso, a arquitetura U-Agro informa o contexto atual do produtor, com as informações sobre o seu perfil e localização.

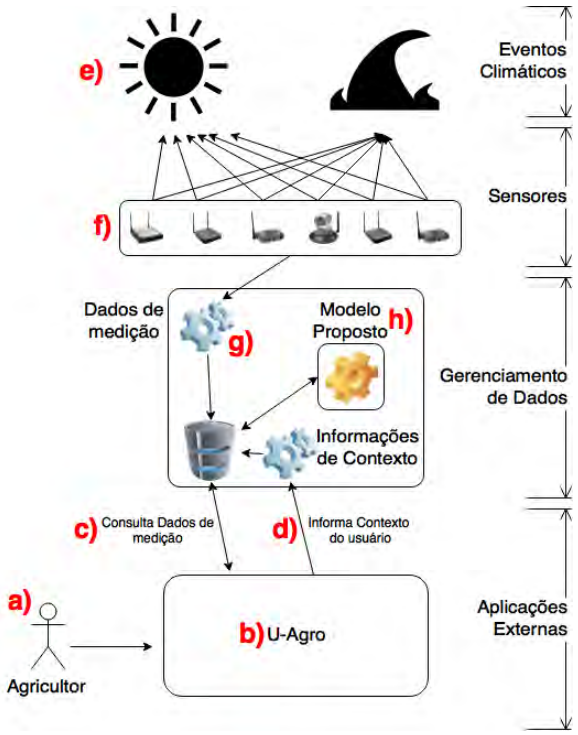


Figura 2 – Visão geral do cenário motivador.

A arquitetura de gerenciamento de observações utiliza o modelo proposto nesta tese (h). No momento da consulta (c), a arquitetura verifica quais informações da consulta estão relacionadas aos elementos de contexto informados (d) e filtra a consulta de informações, retornando apenas informações relacionadas a temperatura ambiente da localização na qual o agricultor se encontra.

(Aldaz et al., 2008) ALDAZ, Eduardo et al. Data Management in the Ubiquitous Meteorological Data Service of the America's Cup. In: 3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008. Springer Berlin Heidelberg, 2009. p. 302-311.

(Hosburgh et al., 2016) HORSBURGH, Jeffery S. et al. Observations Data Model 2: A community information model for spatially discrete Earth observations. Environmental Modelling & Software, v. 79, p. 55-74, 2016.

(Ma et al., 2014) MA, Li et al. A Study of Agricultural Meteorological Monitoring System Based on Wireless Sensor Networks. International Journal of Multimedia and Ubiquitous Engineering, v. 9, n. 7, p. 15-26, 2014.

(Morgenstern et al., 2015) MORGENSTERN, M., ALVES, R., BATTISTI, G., MARAN, V. U-Agro: Uma Arquitetura Ubíqua de Gerenciamento de Atividades na Agricultura de Precisão. Revista Jr de Iniciação Científica em Ciências Exatas e Engenharia - ISSN 2236-0093. N. 10. P.1-8. 2015.

(Serafini, 2008) SERAFINI JÚNIOR, S. Histórico de Instalação das Estações Meteorológicas do INMET no Estado de Minas Gerais. In Simpósio de Pós-Graduação em Geografia do Estado de São Paulo, UNESP – Rio Claro - SP, 2008.

(Staab; Studer, 2013) STAAB, Steffen; STUDER, Rudi (Ed.). Handbook on ontologies. Springer Science & Business Media, 2013.

(Studer et al., 1998) STUDER, Rudi; BENJAMINS, V. Richard; FENSEL, Dieter. Knowledge engineering: principles and methods. Data & knowledge engineering, v. 25, n. 1, p. 161-197, 1998.

(Zink et al., 2010) ZINK, Michael et al. Closed-loop architecture for distributed collaborative adaptive sensing of the atmosphere: meteorological command and control. International Journal of Sensor Networks, v. 7, n. 1-2, p. 4-18, 2010.