Uma Revisão de Técnicas de Distribuição e Persistência de Informações de Contexto e Inferências de Situações em Sistemas Ubíquos

Vinícius Maran 1

José Palazzo M. de Oliveira 1

Resumo: A computação ubíqua (ou computação onipresente) define que a computação alie alta mobilidade e alta imersão computacional no ambiente e na rotina dos usuários. Para prover a imersão computacional necessária, sistemas ubíquos devem ser cientes de contexto. Ou seja, eles devem ser capazes de se adaptar ao ambiente de acordo com o que está acontecendo nele (eventos) ou baseado no seu estado atual. Uma das questões abordadas em pesquisas recentes na área de sensibilidade de contexto trata de formas com que contexto pode ser armazenado e recuperado, e de que forma isto pode ser integrado a aplicações pervasivas para que estas adaptem sua execução ou apresentação de informações ao usuário em tempo real. Neste artigo, apresentamos um estudo sobre as áreas de sensibilidade ao contexto e a situações (uma abstração de alto nível de contextos). A partir deste estudo, e da definição de um estudo de caso baseado na integração entre aplicações de dois domínios, foi possível identificar um conjunto de oportunidades de pesquisa na área.

1 Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil
{vmaran, palazzo@inf.ufrgs.br}

Abstract: *Ubiquitous computing defines that the computing has to combine high mobility and high computational immersion in the environment and in users routine, even those without notice. To provide the required computational immersion, ubiquitous systems must be aware of the context. I.e., they must be able to adapt it selves according to what is happening in the environment (events) or based on environment current state. One of the issues addressed by recent research in context awareness comes to ways in which context can be stored and retrieved, and how these tools can be integrated into pervasive applications so that they adapt their execution or submission of information to the user in real-time. We carried out a study on the areas of context and situation awareness (an abstraction of high-level contexts). From this study, and the definition of a case study based on the integration between two application domains, it was possible to identify a set of research opportunities in the area.*

1 Introdução

Mark Weiser [86] apresentou no artigo *The Computer for the 21st Century* um cenário onde computadores, e a computação em si, se tornariam onipresentes 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 definições surgiu o termo Computação Ubíqua, que define a onipresença da computação perante o usuário e o ambiente. Apesar de ser um tema relativamente antigo, ainda não possuímos casos de implementação plena da computação ubíqua tal qual 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 a aplicação em sistemas considerados ubíquos. Dentre as principais tecnologias e metodologias, podemos citar: (i) Redes de Sensores Sem Fio [34], Internet das Coisas (*Internet of Things* - IoT) [61], Computação Móvel, Computação Pervasiva, Web Semântica [11], e Sensibilidade ao Contexto e a Situções [58].

Neste trabalho, revisamos as áreas de sensibilidade ao contexto e a situações de uma forma geral, apresentando o estado da arte relacionado a estas áreas e de suas sub-áreas. Desta forma, foi possível identificar um conjunto de oportunidades de pesquisa relacionadas as áreas de sensibilidade ao contexto e a situações aplicadas a computação ubíqua.

Esta artigo está organizado da seguinte forma: Na Seção 2 são apresentados os principais conceitos e trabalhos relacionados a área de sensibilidade ao contexto (*Context Awareness* – CA). Na Seção 3 são apresentados os conceitos e trabalhos relacionados a área de computação sensível a situações (*Situation Awareness* – SWA). Na Seção 4, é

apresentado um estudo comparativo de arquiteturas de gerenciamento de contexto e de situações.

Na Seção 5, é apresentado um estudo de caso fictício de um ambiente ubíquo. Através deste estudo de caso, foi possível identificar um conjunto de requisitos necessários para a implementação de arquiteturas ubíquas. Através da comparação destes requisitos e de trabalhos já desenvolvidos, foi possível identificar um conjunto de oportunidades de pesquisa na área.

1.1 Sensibilidade ao Contexto em Sistemas Ubíquos

A área de sensibilidade ao contexto é essencial para a computação ubíqua. 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 [27]. Em arquiteturas ubíquas, a sensibilidade ao contexto pode interferir diretamente em diversas operações, dentre elas podemos destacar [18]:

- **Abstração de Informações de Contexto do Ambiente**: Informações são coletadas do ambiente (geralmente por sensores) 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 contextos dos 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;
- **Reação a Ações Baseada 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;
- **Modificação 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.

2 Context-Aware Computing

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

2.1 Definição de Contexto

Os dados de entrada, coletados por sensores, são abstraídos em conjuntos e então chamados de dados de contexto. Algumas pesquisas buscaram definir o que é contexto. A primeira definição de Contexto foi apresentada por Schilit & Theimer [73], 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.

Alguns anos depois, Dey [27] 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)"*. Além disso, o trabalho de Dey [27] apresentou o conceito de abstração de situação (Seção 3).

Segundo Brézillon & Araújo [15], contexto 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.

Baseado na definição de Dey [27], Zimmermann *et al* [90] definiram uma forma operacional de contexto, separando a definição original em sub categorias de contexto (Figura 1), 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 entidades possam estabelecer com outras entidades).

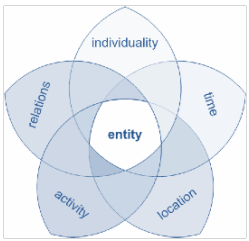


Figura 1: Categorias Fundamentais de Contexto [90]

Além da definição de contexto de uma forma operacional, Zimmermann *et al* [90] 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:

(i) 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 a aproximação dos valores de elementos que juntos, representam o contexto. A Figura 2 apresenta um exemplo de variação da aproximação de valores, onde há variações nos valores de individualidade, localização, tempo e atividade.

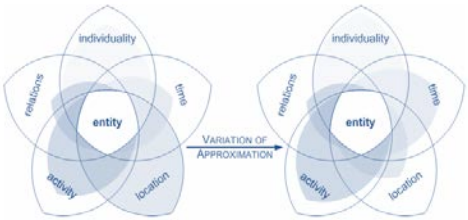


Figura 2: Variação de Aproximação [90]

(ii) 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).

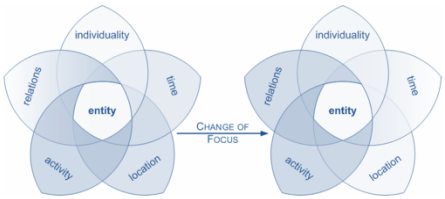


Figura 2: Mudança de Foco [90]

(iii) 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 4.

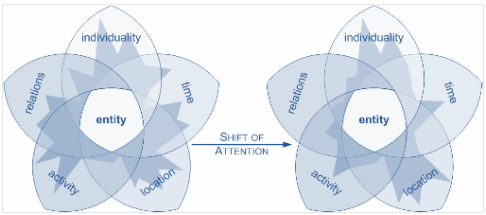


Figura 3: Troca de Atenção [90]

As definições de Dey *et al* [27], Zimmermann *et al* [90] e Brézillon & Araújo [15] são as mais utilizadas em trabalhos que utilizam computação sensível ao contexto. Além disso, a caracterização de contexto tem sido ampliada em pesquisas, como por exemplo na área de computação afetiva, que através de estudos recentes [91][92] buscam incluir a situação emocional de usuários como contexto e tratar este tipo de contexto em sistemas ubíquos.

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 Makris *et al* [54] há claras distinções entre as primeiras definições de contexto (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 [80]. Além disso, há uma falta de separação clara entre os conceitos de contexto, e as informações de contexto [41].

De forma contrária a definição de Dey *et al* [27], que define contexto como a *"caracterização de uma entidade"*, trabalhos recentes [78][54] afirmam que o contexto também pode surgir a partir da atividade geral de um sistema sensível ao contexto, gerando e sustentando um determinado contexto.

Na definição de Dey et al [27], contexto existe somente quando há uma interação entre um usuário e uma aplicação. Porém segundo Makris et al [54], isto não ocorre de fato, e o contexto existe e varia mesmo sem que existam interações entre o usuário e o sistema. 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 1 apresenta um comparativo relacionado as definições de contexto dos principais autores da área.

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 a modelagem de contexto em sistemas ubíquos.

Tabela 1: Características das definições de contexto propostas por alguns autores (adaptado de [54])

[27], [90]	[54], [80]
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

2.2 Representação 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.

2.2.1 Modelos de Representação

- **Modelo Chave-Valor**

O modelo chave-valor é o modelo mais simples de representação de informações de contexto. Consiste de conjuntos de pares com chave-valor.

O modelo de chave-valor foi utilizado nos primeiros trabalhos de representação de contexto [72]. Riva *et al* [67] definiu 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*) [67].

<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 velocidade de consulta). Porém, estes modelos apresentam sérios problemas para representar estruturas complexas de dados e contextos [12][79][61].

- **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*). Alguns trabalhos [39] são extensões de padrões, como o CC/PP (*Composite Capabilities / Preference Profile*) [83] e UAProf (*User Agent Profile*) [4].

Outros trabalhos propuseram extensões da linguagem XML para a representação de contextos. Rukzio et al [71] implementou 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* [1]. A Figura 5 apresenta um exemplo de representação de perfil de usuário, com informações básicas.

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 [69]. A Figura 6 apresenta um exemplo de diagrama CML (a), e a instância de um sensor na conversão para o formato XCML (b).

A ContextML [49] 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).

```
<Profile ...>
<user:User>
<user:Identity>
<user:FirstName>Enrico</user:FirstName>
<user:LastName>Rukzio</user:LastName>
</user:Identity>
<user:Contact>
<user:Email>Enrico.Rukzio@ifi.lmu.de</user:Email>
<user:Homepage>http://www.mimuc.de</user:Homepage>
<user:TelephoneWork>+49 89 2180-4656</user:TelephoneWork>
</user:Contact>
</user:User>
<device:Device>
<device:HardwarePlatform>
<device:ScreenSize>101x80</device:ScreenSize>
<device:Model>T68R1</device:Model>
<device:ImageCapable>true</device:ImageCapable>
<device:Keyboard>PhoneKeypad</device:Keyboard>
<device:Vendor>Ericsson Mobile</device:Vendor>
</device:HardwarePlatform>
</device:Device>
...
</Profile>
```

Figura 4: Representação do perfil de usuário [71]

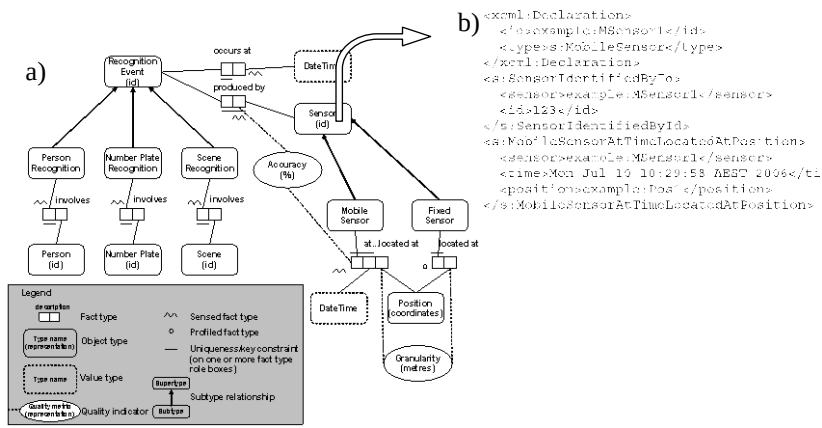


Figura 5: a) Diagrama CML; b) instância de sensor na linguagem XCML (adaptado de [69])

A Figura 7 apresenta um exemplo de representação de um provedor de contexto (localização de um usuário) na linguagem 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>
```

Figura 6: Exemplo de representação de contexto utilizando a ContextML [49]

A CD-XML (*Context-Definition XML*) é uma extensão da linguagem XML para a definição de contextos, e atualmente é utilizada na arquitetura MultiS [30]. Esta linguagem foi desenvolvida para representar mudanças de contexto, que devem modificar a forma de funcionamento da arquitetura MultiS. A linguagem representa as seguintes características:

<sensor>: Representa um sensor que tem a habilidade de monitorar o sistema;

<context>: Representa qualquer item de informação produzido pelo sistema gerenciador de contexto. Cada representação desta *tag* contém uma fonte (que pode ser um <sensor>), e um filtro, definido pela tag <filter>;

<filter>: Representa um filtro para processar uma determinada informação de contexto. Filtros são compostos por parâmetros;

<parameters> e **<parameter>**: <parameters> define um agrupamento de parâmetros, enquanto <parameter> define um parâmetro utilizado por um determinado filtro. A Figura 8 apresenta um exemplo do contexto *"está em um cômodo"* representado na linguagem 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>
```

Figura 7: Exemplo de representação de contexto utilizando a ContextML [30]

Esquemas de marcação tem sido utilizados em muitos sistemas ubíquos para a transmissão de informações e armazenamento temporário de dados. Este modelo é mais estruturado e flexível se comparado a 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 [68][61].

Além disso, dependendo da complexidade dos relacionamentos entre informações estruturadas em esquemas distintos, a recuperação da informação pode se tornar difícil e complexa [61].

- **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* [40], e CML – *Context Modeling Language* [40][42].

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; e, (iii) Representação de fatos temporais; (iv) Representação de qualidade da informação [42]. A Figura 9 apresenta um exemplo de diagrama CML com anotações de qualidade da informação.

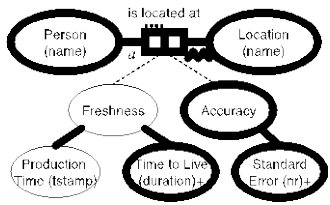


Figura 8: Exemplo de representação de contexto utilizando CML com anotações de qualidade de informações [42]

A ContextUML [75] 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 10.

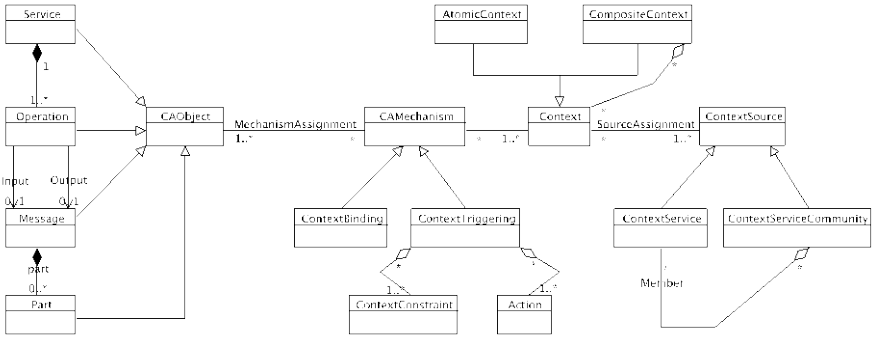


Figura 9: Metamodelo da linguagem ContextUML [75]

Além de extensões da linguagem UML, trabalhos tem sido propostos para a representação gráfica de contextos utilizado DSL (*Domain Specific Languages*). A MLContext [45] propôs uma definição de alta abstração e independente de plataforma para a representação de contextos. A Figura 11 apresenta o metamodelo proposto na MLContext.

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 [79][61].

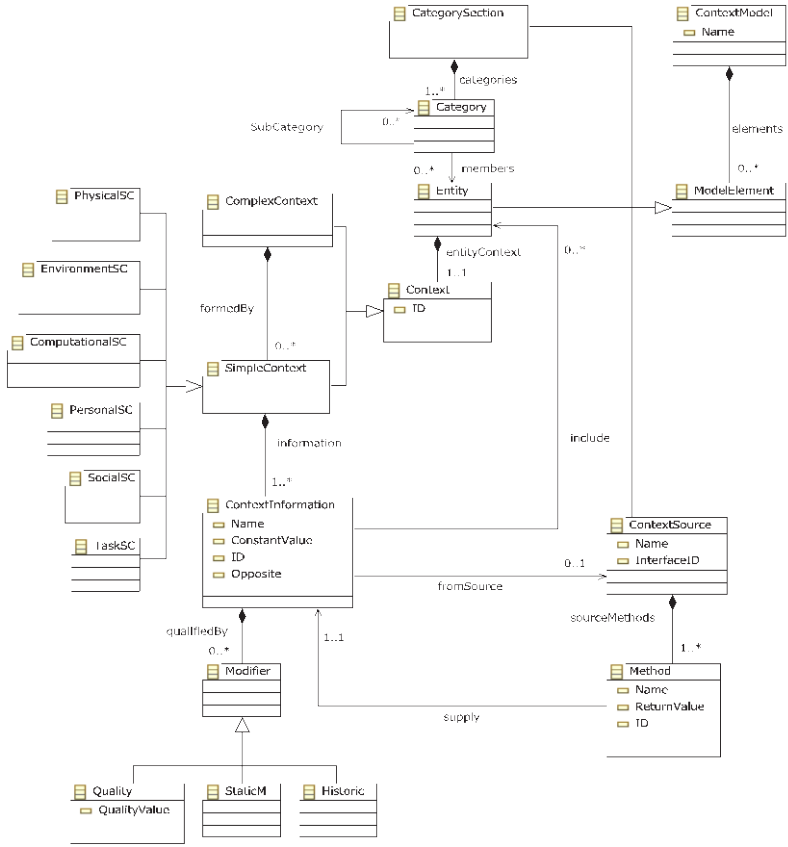


Figura 10: Metamodelo abstrato da linguagem MLContext [45]

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 [79][61].

- **Modelos Orientados a Objetos**

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 [79][61].

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) [61].

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) e traduzidos posteriormente para uma linguagem orientada a objetos [61].

- **Modelos Baseados em Lógica**

Lógica define um conjunto de condições para que uma expressão possa ser resultante, através do processo de inferência [79]. 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 sob regras bem definidas [61].

Além disso, existem problemas de padronização, e isto dificulta a reutilização e aplicabilidade em sistemas ubíquos [61]. Atualmente, são utilizadas em conjunto com outras formas de representação, como por exemplo, ontologias no formato OWL-DL.

- **Modelos Baseados em Ontologias**

De acordo com Borst [14], ontologia pode ser definida como uma especificação formal e explícita de um conjunto de conceitos de forma compartilhada. Como contexto pode ser definido como uma área de conhecimento, nada mais natural que a utilização de ontologias para a representação de informações de contexto.

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 [79].

Existem muitas opções para a realização de inferências em ontologias. Geralmente elas estão associadas a motores de inferência, tais como: Fact++ [82], Racer [81], Pellet [23] e Jess [33]. Porém, a recuperação de contexto pode ter alto custo computacional, 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 [79].

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 aos outros modelos [61][12].

Diversas ontologias tem sido propostas para representar contextos em arquiteturas ubíquas. A ontologia CoOL (*Context Ontology Language*) [93] foi uma das primeiras iniciativas de representação de contexto utilizando ontologias para permitir a interoperabilidade de definições entre sistemas ubíquos.

Este trabalho foi dividido em três partes: (i) Definição de uma ontologia de base (CoOL Core) na linguagem OWL e DAML+OWL; (ii) Definição de regras lógicas na linguagem F-Logic; e (iii) Definição de uma ontologia para descrição de serviços em sistemas ubíquos com a utilização da linguagem DAML-S (Strang et al, 2003). A Figura 12 apresenta a representação de entidades na ontologia CoOL *Core*.

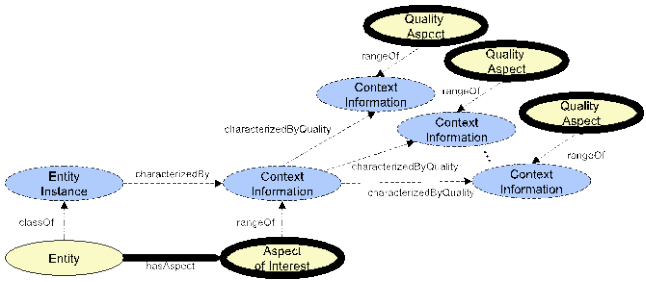


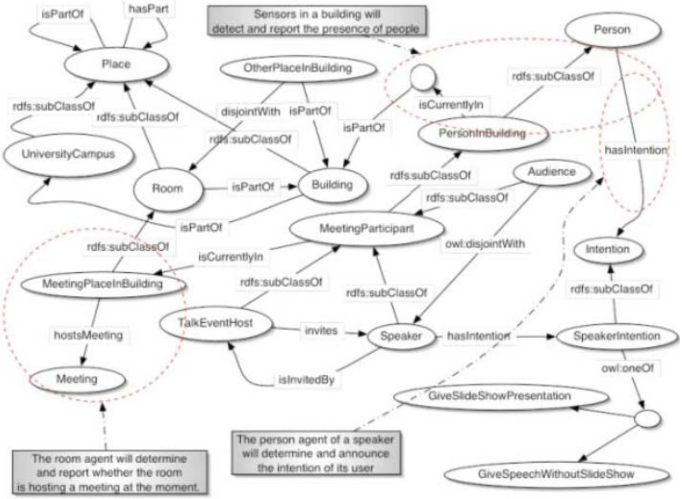
Figura 11: Representação de entidades de contexto e seus relacionamentos na ontologia CoOL *Core* [93]

Além de ontologias de contexto genéricas, existem ontologias que modelam contexto em domínios específicos. A ontologia CoDAMoS [63] modelou contexto relacionado ao domínio de ambientes inteligentes. A ontologia foi definida com base nos principais conceitos relacionados ao contexto de casas inteligentes. São eles: usuário, serviço, plataforma e ambiente. A partir da definição destes conceitos na ontologia, os autores expandiram-nos em sub conceitos e definiram um conjunto de relações entre os conceitos.

A ontologia CoBrA [21] 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 Figura 13 apresenta (a) as principais classes da ontologia CoBrA, e (b) um exemplo do caso de uso *"Presença de usuário"*.

Martins & Silva [56] 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 poder de representação de cada ontologia em relação as regras utilizadas. Baseado na comparação feita por Martins & Silva [56], adicionamos as características de ontologias recentes para a representação de contextos.

a)



b)

T-Box	PessoaNoLocal ≡ Pessoa *and* ∃estáAtualmenteEm.Local
A-Box	Pessoa (João) Local (sala101) estáAtualmenteEm (João, sala101)
Inferido	PessoaNoLocal (João)

Figura 12: a) Principais classes da ontologia CoBrA [21]; b) Exemplo de regra representada na ontologia CoBrA (adaptado de [56])

Como podemos observar na Tabela 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, podemos observar que:

(i) o padrão SWRL foi adotado para a representação de regras para ontologias OWL-DL;

(ii) ontologias recentes utilizam conceitos mais abstratos, se comparadas as primeiras ontologias de representação de contexto, tais como *Situation* e *ContextEntity;*

(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.

O padrão OWL 2 [94] foi proposto como uma atualização do OWL. Porém até o momento não foram encontradas arquiteturas que utilizam os perfis definidos na OWL 2 para a representação de contexto.

Tabela 2: Comparativo entre ontologias em relação a conceitos e expressividade

Conceito/ Ontologia	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Person	√	√	√		√	√		√
Place	√						√	√
Intention	√				√			
User				√				√
Environment				√	√		√	
Platform				√	√			
Service				√	√			
Location		√			√	√		
Activity		√			√		√	
Comp Entity		√			√			
Policy			√					
Action			√				√	
Agent			√					
BDI			√					
Space			√					
Context Entity						√		√
Situation							√	
Expressividade	ALCON	SHI	SHOIN	Taxonomia	SHOIN	SHOIN	SHOIN	SHOIN
Regras	-	SWRL	-	-	SWRL	SWRL	SWRL	SWRL

1: CoBrA [21]; **2:** CONON [84]; **3:** SOUPA [19]; **4:** CoDAMoS [63]; **5:** SmartSpace [46]; **6:** iConAwa [88]; **7:** [7]; **8:** [48]

2.2.2 Comparativos entre Modelos

Strang & Popien [79] realizaram uma comparação entre as formas de representação de contexto mais utilizadas em sistemas ubíquos. Para realizar esta comparação, eles elencaram 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*) [79]. O resultado do comparativo é apresentado na Tabela 3.

Como podemos observar na tabela resultante do comparativo, 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: Comparativo entre formas de representação de contexto [79]

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	++	++	+	+	++	+

De acordo com Bettini et al [12], a modelagem de contextos e sistemas de gerenciamento de contexto em geral possuem os seguintes requisitos:

(i) **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;

(ii) **Relacionamentos e Dependências:** Existem 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;

(iii) **Sem restrição de tempo**: Aplicações sensíveis ao contexto precisam 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;

(iv) **Imperfeição**: Contexto é gerado a partir de diversas fontes. Em muitos casos, a informação destes sensores são coletadas os distribuídas 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;

(v) **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;

(vi) **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;

(v) **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 4 apresenta uma comparação entre três formas de modelagem de contexto.

Tabela 4: Comparativo entre formas de modelagem de contexto [12]

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

+ Satisfatório
~ Parcialmente satisfatório
- Insatisfatório

Como podemos observar na comparação, Modelos Ontológicos possuem maior capacidade de representação e inferência, porém são pouco eficientes. Esta baixa eficiência se deve principalmente a complexidade computacional envolvida na construção de motores de inferência [12].

Os modelos espaciais são mais eficientes, porém não possuem tanta capacidade de representação em comparação aos modelos ontológicos e baseados em objetos. Como podemos observar na Tabela 4, nenhuma das formas analisadas foi satisfatória na questão de imperfeição da informação.

Este problema pode 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 [12].

2.3 Consulta de Contexto e Recuperação de Informação Baseada em Contexto

Informações de contexto são essenciais para sistemas ubíquos, pois o tratamento destas informações e a sua utilização permitem com que sistemas ubíquos possam se adaptar as necessidades dos usuários e de outros sistemas. Estas adaptações devem ser feitas em tempo real, e podem ser resultar tanto em mudança de comportamento das aplicações (execução), quanto em recuperação de informação (conteúdo) [95]. Informação pode ser definida como *"uma coleção de documentos discretos, onde cada documento pode ser subdividido em um conjunto de campos"* [16].

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 [16]. Portanto, contexto pode ser utilizado de duas formas na recuperação de informação [16]:

- *Para derivar uma consulta que retorne os documentos que melhor se enquadrem no contexto requisitado;*

- *Para tratar contexto como um documento, ou seja, o contexto se torna a fonte de informação a ser consultada.*

A área de CAR envolve trabalhos que contemplam, desde a modelagem e representação de consultas e documentos, até arquiteturas de suporte a recuperação de informação de forma contextualizada.

Chi *et al* [97] introduziu o conceito do relacionamento *in* em consultas baseadas em contexto. Desta forma, um termo pode ser localizado de acordo com o contexto sob o qual é utilizado. A seguir é apresentado um exemplo de aplicação do relacionamento binário *in*.

<document_term> in <context>
nota in supermercado

No exemplo apresentado, *nota* é relacionado ao contexto supermercado no momento da consulta, pois nota pode ter diferentes significados em diferentes contextos (por exemplo, nota em uma universidade ou nota em um jornal) e consequentemente, a busca pode retornar documentos que não são relevantes.

Haghighi *et al* [38] apresentou um comparativo de linguagens de consulta de contexto. O comparativo comparou soluções propostas de acordo com modelos de representação (XML, ontologias, modelo relacional, entre outros) com os seguintes requisitos de consulta de contexto:

- **Estático ou Dinâmico**: Determinadas informações de contexto podem ser estáticas, como por exemplo o perfil de um usuário, ou podem ser dinâmicas, como por exemplo a localização de um usuário ou recurso;

- **Fluxo de Dados Contínuo**: Informações de Contexto podem ser transmitidas como um fluxo contínuo de dados (como por exemplo leituras contínuas de sensores);

- **Metadados e Qualidade do Contexto (*Quality of Context* - QoC)**: Informações de contexto podem ser associadas a metadados. Além disso, informações de contexto podem ser temporais e são propensas a imperfeição, erros, ambiguidade ou podem ser incompletas;

- **Proximidade e Relacionamentos Espaciais**: Informações de contexto podem expressar proximidade e podem representar dimensões como latitude, longitude e altitude;

- **Situações**: Informações de contexto podem ser inferidas e derivadas para representar situações.

Os autores concluíram com base no comparativo que as linguagens de consulta baseadas em SQL e RDF são as que conseguem atender aos principais requisitos para

consultas de contextos. As linguagens baseadas em RDF não possuem suporte a fluxo contínuo de informação e as linguagens baseadas em SQL precisam de grandes extensões e implementação de tradutores para atender aos requisitos de consulta de contextos.

O framework **HyConSC** [5] cataloga e recupera contextos. A catalogação de documentos e de contextos como documentos foi feita com base em um conjunto de *templates*. A partir da utilização dos *templates* em documentos, foi possível realizar consultas baseadas em três tipos de contextos: Físico, Digital e Conceitual. A Figura 14 apresenta a interface de consulta utilizada pelo framework.

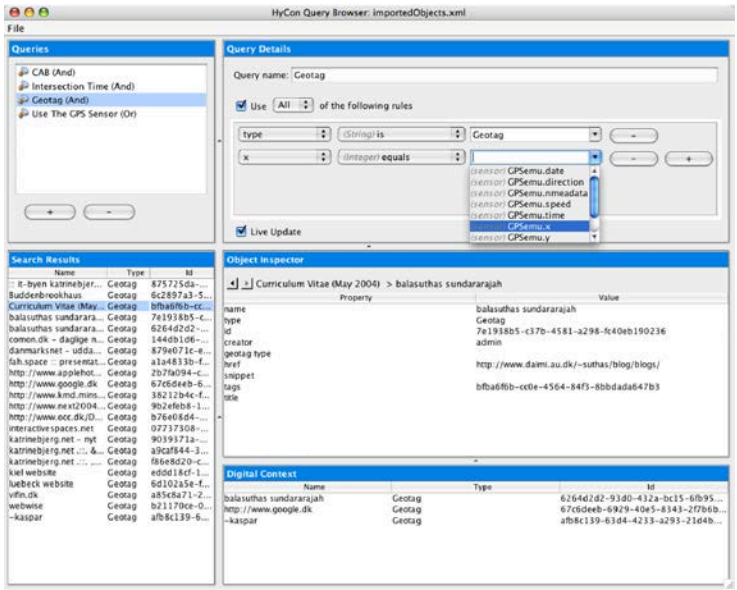


Figura 13: Interface de consulta baseada no framework HyConSC [5]

A **CQL** (*Context Query Language*) [66] é baseada em XML, e permite a consulta de contextos modelados em ontologias. A linguagem define um conjunto de operadores para o tratamento de contextos. A Figura 15 apresenta um exemplo de consulta de contexto, onde deseja-se saber os endereços de *Mary* na *Itália*. Apesar de apresentar os principais conceitos da linguagem, os autores não apresentam a implementação de um protótipo para o processamento da linguagem. Frá et al [32] estendeu a CQL, e a partir da implementação de um framework, incorporou a CQL em dois projetos europeus.

A **f-SPARQL** [24] é uma extensão da linguagem SPARQL para suportar o reconhecimento de contexto. Ela foi implementada para integrar consultas em definições feitas em RDF e classificações Fuzzy para tratar a imperfeição e classificar informações brutas de sensores.

```
<ctxQuery resultName="addressOfMaryInItaly">
  <entity  ontConcept="prefix:music:username">
  user|Mary
  </entity>
  <scope
      ontConcept="prefix:music:CivilAddress"
      ontRep="prefix:music:DefaultAddressRep">
      civilAddress
  </scope>
  <action type="SELECT"/>
    <conds>
      <cond type="ONVALUE">
       <constraint par="civilAddress.country"
                   op="EQ" value="Italy"/>
      </cond></conds></action>
</ctxQuery>
```

Figura 14: Exemplo de consulta na CQL proposta por Reichle et al [66]

A **CARVE** [13] é uma metodologia de geração automática de visões em bancos de dados relacionais a partir de taxonomias que representam informações de contexto (Figura 16). A partir desta metodologia, foi possível filtrar informações de bancos de dados relacionais a partir de uma lista de termos que representaram contextos. Porém, ainda não foram feitas definições para a consulta baseada em contextos e a ligação da taxonomia criada com outras modelagens de contexto (por exemplo, com ontologias).

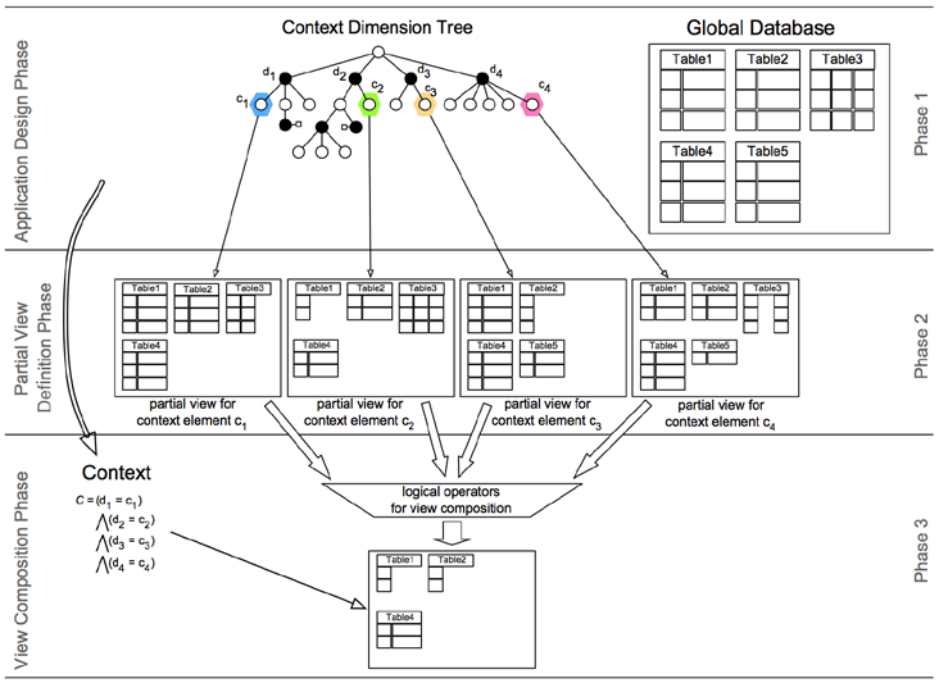


Figura 15: Metodologia CARVE [13]

O **PerLa** (*Pervasive Language*) [73] é um middleware de gerenciamento de informações de contexto. Ele implementa a linguagem *PerLa*, que define uma extensão de SQL e pode ser descrita em XML. A Figura 17 apresenta um exemplo de consulta na linguagem PerLa. No exemplo apresentado, são calculados todos os dispositivos de baixa potencia registrados no ambiente, a cada 24 horas. O middleware *PerLa* permite com que consultas sejam executadas de forma distribuída entre os nós de uma rede distribuída, e concentrados em um servidor. Por exemplo, consultar a temperatura média em todos os sensores de temperatura de uma casa.

```
CREATE OUTPUT STREAM NumberOfLowPoweredDevices (counter INTEGER)
AS HIGH:
    EVERY 24 h
    SELECT COUNT(*)
    FROM LowPoweredDevices(24 h)
```

Figura 16: Exemplo de consulta na linguagem PerLa [73]

2.4 Persistência de Contexto

Como visto anteriormente, ontologias tem sido amplamente utilizadas para a representação de contextos e tem-se demonstrado como a melhor forma de representação de acordo com os estudos comparativos feitos recentemente. Desta forma, analisamos as soluções de persistência de ontologias, como soluções de persistência de contexto.

Metodologias de integração entre ontologias e modelos de bancos de dados tem sido propostas em duas linhas distintas: (i) integração de ontologias em bancos de dados relacionais – primeiro tipo de integração entre ontologias e bancos de dados [89]; e (ii) integração de ontologias e bancos de dados *NoSQL* (*Not Only SQL*) ou uso de sistemas de arquivos distribuídos.

Dados semânticos são utilizados em muitas áreas da computação. Portanto, algumas soluções são definidas com propósitos específicos, com linguagens de representação e consulta específicos para o domínio onde serão aplicadas. As informações das metodologias pesquisadas neste trabalho são apresentadas na Tabela 5.

2.4.1 Bancos de Dados Relacionais

Muitos trabalhos relacionados a metodologias de integração entre bases de dados e ontologias utilizam o modelo relacional como base para a integração. Este fato acontece principalmente pelo fato de que bancos de dados relacionais são utilizados em diversos domínios e pelo formalismo no qual está baseado.

O *SciSPARQL* [6] é uma arquitetura de expansão da linguagem SPARQL para tratar de questões pertinentes ao domínio científico – especificamente em grupos de pesquisas que utilizam informações modeladas no formato RDF e possuem grandes quantidades de informações geradas. Esta arquitetura permite que dados científicos sejam armazenados no formato de triplas RDF utilizando o modelo de dados relacional.

Para realizar a consulta de informações, a arquitetura permite o uso da SciSPARQL, uma linguagem de consulta que amplia as funções da linguagem de consulta SPARQL com funções externas definidas na linguagem *Python*. O *SciSPARQL* não suporta a realização de inferências nas ontologias persistidas. Tal característica é citada como um tópico de trabalhos futuros pelos autores do trabalho.

O trabalho apresentado por Shan *et al* [74] apresenta uma metodologia de integração entre ontologias definidas no formato RDF e persistidas nos bancos de dados relacionais *MySQL* ou *PostgreSQL*. Através de uma tabela de mapeamento criada especificamente para esta arquitetura, foi utilizado o algorítmo C-Store [74] para a definição do modelo de banco de dados baseado nas ontologias inseridas. Foi utilizada a linguagem SPARQL para a consulta de informações persistidas no banco de dados.

NYAYA [25] é uma arquitetura de persistência de dados semânticos – representados em RDF ou OWL-DL; que permite que grandes quantidades de informação sejam armazenadas. Para fazer isto, cada RDF importado na arquitetura é convertido em uma estrutura chamada *Semantic Data Kiosks*, com estruturas e dados modelados de acordo com o RDF importado.

Estas estruturas também geram alguns dados extras, que permitem que consultas sejam feitas utilizando uma linguagem própria, baseada em *Datalog*. Quando uma consulta nesta linguagem é realizada, a arquitetura converte a consulta em subconsultas na linguagem SQL, e então acessa os dados persistidos no banco de dados relacional.

O ERMOS [76] é um projeto que implementa a integração entre ontologias OWL-DL e o banco de dados relacional *MySQL*. Ele implementa a linguagem *Rule based Ontology Query Language* (ROQL). Através desta linguagem, os desenvolvedores podem realizar consultas (porém sem suporte a inferências), que posteriormente são convertidas em instruções SQL.

Tabela 5: Informações sobre ferramentas de integração entre ontologias e bancos de dados

Trabalho	**SciSPARQL [6]**	**OXDBS [59]**	**JenaPro [85]**	**[74]**	**Swarms [26]**	**NYAYA [25]**	**OCSS [3]**	**ERMOS [76]**	**Db4 OWL [9]**	**Semantic Couch [55]**
Modelo de Dados	*RDBMS*	XML	Arquivos *Hadoop*	*RDBMS*	Arquivos	*RDBMS*	Arquivos	*RDBMS*	Orientado a Objetos	JSON
Linguagem de Consulta	Sci SPARQL	nRQL	Não Menciona	Sparql	Alc	Própria, baseada em *Datalog*	Não Menciona	ROQL	RuleML	QBE
Formato Suportado	RDF	RDF / OWL-DL	RDF	RDF	RDF	RDF / OWL-DL	RDF / OWL	RDF / OWL-DL	RDF / OWL-DL	RDF / OWL-DL
Inferências	Não	Sim, *RacerPro*	Não	Não	Sim	Sim	Não	Não	Sim	Não
Sistema Utilizado	Não Menciona	*eXist XML Database*	*Hadoop*	*PostgreSQL / MySQL*	Não Menciona	Não Menciona	*GFS*	*MySQL*	*Db4Owl*	*CouchDB*

2.4.2 Bancos de Dados NoSQL

Baseado na possibilidade de inserção de arquivos XML sem a necessidade de conversão para outros modelo de dados, o OXDBS [59] propôs a extensão de funcionalidades do banco de dados *eXist* para a possibilidade de verificação de sintaxe de arquivos RDF e OWL-DL, além de permitir o uso do motor de inferência *RacerPro* para consulta e inferência de ontologias.

A arquitetura db4OWL [9] propôs a integração entre ontologias OWL-DL e o banco de dados orientado a objetos *DB4O*. Para realizar esta integração, os autores criaram uma modelagem de classes na linguagem Java que representa a estrutura encontrada em arquivos OWL. A partir disto, esta estrutura e as informações nela contidas foram armazenadas no banco de dados. As consultas e inferências são escritas no formato RuleML [9], e posteriormente convertidas para o formato *Query by Example*, utilizado pelo banco de dados *DB4O*.

A arquitetura *SemantiCouch* [55] propôs a integração entre ontologias OWL-DL e o banco de dados orientado a documentos *CouchDB*. Desta forma, as informações poderiam ser replicadas de forma mais facilitada se comparada a outras abordagens. Para realizar a consulta de dados, a arquitetura oferece uma API baseada no princípio de *Query by Example* (consulta utilizando exemplos).

Outras abordagens propõem a integração de ontologias e sistemas de arquivos distribuídos, permitindo que a mesma definição ontológica seja replicada em diversos lugares.

A ferramenta *JenaPro* [85] apresenta uma arquitetura de integração entre o *framework Jena* – para a utilização de ontologias na linguagem *Java*; e a arquitetura *Hadoop* – utilizada para a distribuição de informações. O trabalho não menciona se implementa alguma forma de consulta ou inferência sob as ontogias persistidas, apenas que estes são tópicos para trabalhos futuros.

O projeto *Swarms* [26] apresenta uma arquitetura para a persistência de dados semânticos representados nos formatos RDF. A arquitetura foi modelada e implementada de forma que permite a distribuição e replicação de informações através de nós de uma rede. Esta distribuição de informação é feita de forma inteligente, através da análise de caminhos entre os nós, para escolher a melhor rota de distribuição da informação. Para realizar a consulta de informações, os autores utilizam a linguagem *ALC* [26].

O OCSS (*Ontology Cloud Storage System*) [3] apresenta uma arquitetura e um conjunto de algorítmos de leitura e escrita em ontologias modeladas nas linguagens RDF e OWL persistidas no sistemas de arquivos GFS (*Google File System*). Esta arquitetura permite que uma infra estrutura de *Cloud Computing* seja criada para o armazenamento e consulta de ontologias.

A Tabela 6 apresenta o resultado da relação entre os principais requisitos para a persistência de ontologias e metodologias recentes de integração entre ontologias e bancos de dados.

O requisito *baixo overhead na conversão de dados* é contemplado nos trabalhos mais recentes. Nos trabalhos [85][26][3] não há conversão de arquivos, pois os trabalhos propõem

arquiteturas de persistência utilizando sistemas de arquivos distribuídos. O trabalho de Neumann et al [59] também não apresenta conversão de arquivos, pois utiliza XML como modelo de dados para armazenamento.

Tabela 6: Comparativo entre ferramentas de integração de ontologias com bancos de dados

Trabalho	**[6]**	**[59]**	**[85]**	**[74]**	**[26]**	**[25]**	**[3]**	**[76]**	**[9]**	**[55]**
Baixo *overhead* na conversão de dados	-	✓	✓	-	✓	-	✓	-	✓	✓
Facilmente Escalável Horizontalmente	-	-	✓	-	✓	-	✓	-	-	✓
Suporte a linguagens de consulta / inferência para OWL (*SWRL*, *SQWRL* ou baseadas em *Datalog*)	-	-	-	-	✓	✓	-	-	✓	-
Suporte a Inferências e Modelagem temporal de Situações	-	-	-	-	-	✓	-	-	✓	-
Integração com Plataformas Móveis	-	-	-	-	-	-	-	-	-	-
Suporte ao padrão OWL-DL	-	✓	✓	-	-	✓	✓	✓	✓	✓

Outros trabalhos recentes [9][55] propuseram a utilização de modelos de armazenamento que diminuíram a criação de novos dados no processo de persistência de ontologias. Especificamente nestes dois trabalhos, foram utilizados os modelos orientados a objetos, e baseados na linguagem JSON.

Em relação à escalabilidade horizontal, apenas 4 trabalhos do comparativo ofereceram alguma solução. Os trabalhos [85][26][3] oferecem escalabilidade horizontal através da utilização de sistemas de arquivos distribuídos. Porém, estes não utilizam um sistema gerenciador de banco de dados, desta forma, não oferecem linguagens de consulta ou de inferência para ontologias. Isto pode ser contornado com a utilização de um motor de inferência externo. O trabalho [55] propôs a utilização de um banco de dados facilmente escalável, apresentando testes de replicação e integração de informações em nós distribuídos.

Em relação às linguagens de consulta, a maioria dos trabalhos não permitem que os sistemas façam consultas e inferências utilizando linguagens baseadas em Datalog. Na maioria destes trabalhos, é utilizada a linguagem SPARQL – padrão definido pela W3C para consulta de documentos RDF em bancos de dados.

Porém a linguagem SPARQL não oferece meios de realização a inferências, suportadas pelas linguagens SWRL e SQWRL, tão pouco oferece suporte a construtores e definições importantes encontradas no padrão OWL-DL.

Ainda em relação às linguagens de consulta, sem a possibilidade da realização de inferências nos dados armazenados nas bases de dados, alguns trabalhos não oferecem a possibilidade de consultas e inferências de situações baseadas em questões de tempo. A maioria dos trabalhos recentes suporta a persistência de arquivos no padrão OWL-DL, seja totalmente, seja parcialmente. Isto permite que contextos sejam representados e armazenados

por arquiteturas ubíquas. Porém, nenhum dos trabalhos pesquisados apresentou a integração em forma de testes com plataformas móveis.

Este requisito é importante em arquiteturas ubíquas, pois diminui a necessidade de existência de um servidor central com todas as definições de representação de contexto. Ao invés disto, dispositivos móveis podem conter e manipular a definição de contexto de acordo com o dispositivo, e posteriormente apenas replicam o contexto captado nestes dispositivos.

3 Situation-Aware Computing

A definição de contexto é amplamente utilizada em sistemas ubíquos. Porém, mais recentemente, a definição de computação sensível a situação também tem sido utilizada constantemente.

3.1 Definição de 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 [12]. Isto permite com que especificações de alto nível sobre atividades humanas sejam utilizadas em compartilhadas em sistemas ubíquos.

De acordo com Endsley [29] sensibilidade a situação (*situation awareness* – SWA) pode ser definida como *"a percepção sobre elementos do ambiente associados a localização e tempo, a compreensão sobre o significado destes elementos e a projeção do estado destes elementos em um futuro próximo"*.

A Figura 18 apresenta os níveis de abstração de contexto utilizados em arquiteturas sensíveis ao contexto. 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.

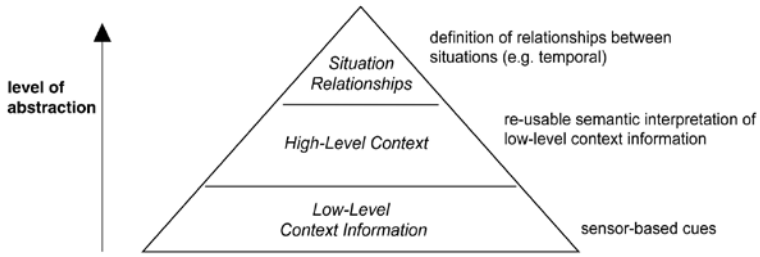


Figura 18: Níveis de abstração de informações de contexto [12]

A partir deste nível, as informações representadas são passíveis de reutilização por outros domínios. O terceiro nível (*Situation Relationships*) 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 19, 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.

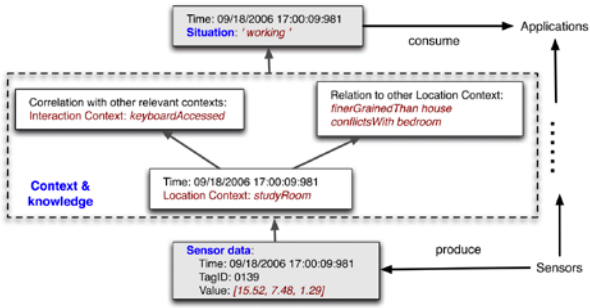


Figura 19: Exemplo de fluxo de informação em sistemas pervasivos [96]

Situações possuem ligações semânticas ricas entre si, tais como [96]:

- **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;
- **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;
- **Sequencia 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, abordamos apenas aspectos relacionados a representação e especificação de situações baseados em ontologias.

3.2 Representação e Especificação de Situações em Ontologias

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 [12]. 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 [96].

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* (adaptada de [36]).

(?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 e Retschitzegger [10] realizaram um comparativo entre os principais trabalhos de ontologias para a representação de situações [57][87][20][84] com base em alguns critérios importantes. São eles [10]: **(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 [20] 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 [10] são:

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

4 Arquiteturas de Gerenciamento de Contexto

Arquiteturas de gerenciamento de contexto tem sido propostas com a utilização de diversas tecnologias e modelos. Estas tecnologias e modelos são variantes nas diferentes ações realizadas em arquiteturas baseadas em contexto.

4.1 Fundamentos

Os procedimentos gerenciados em arquiteturas baseadas em contexto podem ser resumidas em (Figura 20) [54]:

- **Aquisição de Informações de Contexto**: É a funcionalidade mais utilizada em arquiteturas baseadas em contexto. Trata transformação das coletas de informações do ambiente (dados brutos) em informações de contexto de baixo nível. Para que uma arquitetura de contexto atenda aos requisitos necessários de aquisição de contexto, ela deve obrigatoriamente: (i) fornecer maneiras fáceis de implementar a aquisição, (ii) fornecer maneiras fáceis de utilização, e (iii) fornecer métodos de aquisição não intrusivos aos usuários [64].

- **Modelagem de Contexto**: Informações de contexto podem ser modeladas de várias formas. Estas formas estão diretamente ligadas a expressividade de informações de contexto, as formas de raciocínio possíveis, e as formas de persistência e recuperação de contextos.

- **Troca de Informações**: A troca de informações de contexto entre módulos das arquiteturas baseadas em contexto é fundamental para o funcionamento destas arquiteturas. Os requisitos necessários para a implementação de uma troca efetiva de informações de contexto são: (i) escalabilidade, (ii) interoperabilidade, (iii) adaptabilidade, (iv) tratamento de informações de contexto contraditórias, (v) implementação de disseminação de informações baseada em conteúdo, (vi) independência de aplicação, e (vii) utilização de protocolos de comunicação leves [54].

- **Avaliação da Informação**: Através das funcionalidades apresentadas anteriormente, considera-se que os atores (entidades do sistema) possuem todas as informações de contexto necessárias para a execução de processos. Assim, faz-se necessária a tomada de decisão de execução de processos ou apresentação de conteúdo de acordo com as informações de contexto coletadas. Para fornecer uma avaliação eficiente baseada nas informações coletadas, as arquiteturas baseadas em contexto devem: (i) realizar um tratamento eficiente em relação as mudanças dinâmicas do ambiente e as situações inesperadas, (ii) garantir o gerenciamento otimizado do sistema em relação aos recursos utilizados, (iii) manter o nível de satisfação dos usuários mais alto do que o previsto pelos

analistas do sistema, (iv) gerenciar problemas de otimização ocorridos pela necessidade de tomada de decisão em tempo real, e (v) minimizar a intervenção humana [54].

- **Lógica de Negócio**: A lógica de negócio de arquiteturas baseadas em contexto está diretamente relacionada aos processos finais desta arquitetura. Entendem-se como processos finais os processos descritos em alto nível e que interferem diretamente no funcionamento do sistema em relação ao usuário.

- **Funcionalidades Horizontais**: Funcionalidades horizontais são definidas como um conjunto de características desejáveis na execução de todos os processos de arquiteturas baseadas em contexto. Estas funcionalidades são divididas em três categorias, nomeadas: (i) Segurança, (ii) Privacidade, e (iii) Confiança [54].

Figura 20: Principais funcionalidades de arquiteturas baseadas em contexto (adaptado de [54])

Arquiteturas e middlewares tem sido propostos para a manutenção de informações de contexto e como formas facilitadoras de utilização de contexto em diversos domínios. Usualmente, arquiteturas de contexto possuem duas formas de construção: **(i)** arquiteturas de gerenciamento de contexto genérico, que podem ser utilizadas em domínios distintos, e **(ii)** arquiteturas de contexto específicas de domínio, que são construídas com ferramentas que permitem a integração com dados e processos específicos de um determinado domínio.

4.2 Arquiteturas de Gerenciamento de Contexto

Trabalhos recentes [61][54][44] realizaram comparativos entre arquiteturas baseadas em contexto, de acordo com suas características, funcionalidades e atendimento aos requisitos desejáveis para a utilização de contextos em sistemas computacionais.

O ***Context Toolkit*** [27] foi um dos primeiros frameworks conceituais propostos para o desenvolvimento de aplicações baseadas em contexto. De acordo com o trabalho, arquiteturas sensíveis ao contexto devem atender a seis requisitos básicos:

(i) **Separação de Interesses**: Arquiteturas sensíveis ao contexto devem abstrair informações de sensores em relação a lógica de programação das aplicações,

(ii) **Interpretação de Contexto**: Informações coletadas por sensores devem ser tratadas e interpretadas pelas arquiteturas como conhecimento,

(iii) **Comunicação Distribuída e Transparente**: A comunicação entre os componentes de arquiteturas sensíveis ao contexto devem ser transparentes do ponto de vista de aplicações externas e os componentes devem estar distribuídos,

(iv) **Aquisição de Contexto Constante**: A coleta de informações de contexto deve ser constante e não deve ter interrupções,

(v) **Persistência e histórico de contexto**: Informações de contexto devem ser armazenadas e um histórico destas informações deve ser mantido;

(vi) **Descoberta de recursos**: Arquiteturas de gerenciamento de contexto devem ser capazes de localizar recursos disponíveis no ambiente.

O framework conceitual oferece um conjunto de abstrações e ferramentas que permitem o suporte a informações de contexto. Dentre as três principais abstrações criadas no framework, estão: **(i)** *Context Widget*: permite a representação de baixo nível dos dados coletados em sensores, **(ii)** *Context Interpreter*: oferece um conjunto de abstrações e ferramentas para a realização de inferências baseadas em dados de contexto, e **(iii)** *Context Aggregator*: Modela funções de agregação de informações de contexto [27][61].

O middleware ***Aura*** [35] foi definido como um middleware de integração do espaço dos usuários entre diversos dispositivos. Desta forma, os dois principais objetivos deste middleware eram: (i) Permitir a continuidade do trabalho dos usuários, mesmo que os usuários utilizem diversos dispositivos do ambiente, e (ii) Adaptar a utilização de recursos do sistema de acordo com a inclusão ou remoção dinâmica de recursos no ambiente.

O middleware é composto por quatro componentes principais: (i) *Context Observer*: coleta informações de contexto dos *Context Providers* e as envia para os gerenciadores de eventos, (ii) *Task Manager*: gerencia a execução de quatro tarefas pré definidas no middleware, (iii) *Environment Manager*: gerencia as fontes de contexto e componentes do middleware, e (iv) *Context Provider*: coleta as informações de contexto do ambiente. O middleware foi testado em conjunto com duas aplicações pervasivas. As duas aplicações utilizaram dados de contexto gerenciados pela arquitetura para a tomada de decisão (com base na localização e perfil do usuário).

Gaia [70] foi uma das primeiras arquiteturas que utilizam informações de contexto modeladas em ontologias. Em conjunto com a arquitetura, foi proposto um framework de inferência baseado em Prolog para a inferência de contextos baseado na execução de regras lógicas. A principal contribuição da arquitetura *Gaia* não foi a criação de serviços distintos para o gerenciamento de contexto, mas sim a possibilidade de criação de um espaço (formado por conjuntos de *softwares* e *hardwares*) que possibilitaram a utilização de informações de contexto.

A arquitetura é composta por seis componentes principais: (i) *Context Provider*: Adquire dados de contexto de sensores e outras fontes, (ii) *Context Consumer*: Aplicações que podem estar interessadas no contexto do ambiente, (iii) *Context Synthesiser*: Agrega informações de baixo nível de contexto e as transforma em informações de alto nível, (iv) *Context Provider Lookup Service*: Mantém um registro atualizado de fontes de informação de contexto no ambiente, (v) *Context History Service*: Armazena e recupera o histórico de contexto, e (vi) *Ontology Server*: Armazena e gerencia as ontologias utilizadas na arquitetura.

CoBrA (*Context Broker Architecture*) [20] é uma arquitetura baseada em agentes que fornece ferramentas para o compartilhamento de conhecimento e inferência sobre contextos em ambientes inteligentes. A arquitetura utiliza ontologias OWL e RDF para a representação de informações de contexto e trabalha de forma centralizada. A interação com ontologias é feita utilizando o framework Jena [17].

Para a persistência de informações, a arquitetura utilizou as ferramentas providas pelo framework Jena. As informações trafegam entre os componentes da arquitetura e sensores em RDF em conjunto com o padrão SOAP.

A ***CROCO*** (*CROss application COntext management*) [62] é uma arquitetura baseada em um conjunto de serviços baseados em ontologias para a modelagem e gerenciamento de contextos. A arquitetura tem como objetivo ser responsável por três funcionalidades: (i) gerenciamento dos dados de contexto, através da persistência de dados inferidos para a utilização de informações históricas, (ii) verificação da consistência e inferência de contextos, e (iii) fornecimento de ferramentas para a atualização de informações de contexto.

Através da definição da ontologia *CROCOON* (*Cross-application Context On- tology*) [62], a arquitetura permitiu com que fossem feitas consultas utilizando a linguagem SPARQL. A ontologia CROCOON utilizou conceitos comuns de ontologias conhecidas da área, como a SOUPA [20] e OWL-Time [60].

Hydra [8] é um middleware que tem como objetivo integrar conceitos provenientes da área de IoT (*Internet of Things*) no gerenciamento de sistemas inteligentes baseados em contexto. Este middleware utiliza modelagens baseadas em chave-valor e orientação a objetos para modelar informações de contexto.

Para realizar a inferência sobre informações de contexto, o middleware utiliza a ferramenta Drools, baseado em regras lógicas. Para a persistência de contexto, o Hydra serializa os objetos utilizados para representar contextos em documentos XML e os armazena remotamente.

O ***Feel@Home*** [37] é um framework de gerenciamento de contexto que permite a interação entre diferentes domínios de aplicação. As informações de contexto são modeladas em ontologias OWL. A modelagem de contexto é gerenciada por módulos (*managers*), separados em três categorias, de acordo com o domínio: (i) Gerenciadores de contexto de casas, (ii) Gerenciadores de contexto de Escritórios, e (iii) Gerenciadores móveis de contexto.

Baseado no conceito de gerenciadores separados por domínio, o framework permite a interação de componentes de duas formas: (i) dentro do mesmo domínio, ou (ii) entre domínios distintos. Os dados modelados em OWL são armazenados utilizando o framework Jena, e são distribuídos entre os gerenciadores da arquitetura utilizando o padrão SOAP.

Ainda na área de integração entre conceitos da área de IoT e gerenciamento de contexto, o sistema ***Octopus*** [31] foi definido como uma ferramenta extensível e open source para suportar a coleta e fusão de informações para aplicações que desejam suportar as funcionalidades da área de IoT. O principal objetivo deste sistema é permitir que leigos em sistemas baseados em contexto consigam integrar informações provenientes de sensores em suas aplicações. Para fazer isto, o sistema oferece um conjunto de abstrações e ferramentas de gerenciamento, que permitem a realização do cadastro e do gerenciamento de sensores.

C-CAST [65] é um middleware que propõe a integração de redes de sensores sem fio (*Wireless Sensor Network* - WSN) em arquiteturas baseadas em contexto. A arquitetura é dividida em tres camadas, nomeadas: **(i)** Provedores de Contexto, **(ii)** Gerenciadores de contexto, e **(iii)** Disseminadores de contexto. A arquitetura não utiliza ontologias para a representação de contexto, com a justificativa de que ontologias consomem muitos recursos computacionais e impedem a sua utilização em tempo real para a inferência de contextos

A ***FOCALE*** [47] é uma arquitetura de gerenciamento de recursos e de informações de contexto. O gerenciamento destes recursos é realizado com a utilização de informações de contexto, que determinam a ocorrência de eventos e consequentemente, a necessidade de realocação de componentes, serviços e recursos. A arquitetura (Figura 21) utiliza ontologias e modelagem orientada a objetos para representar contextos em alto nível. Para o gerenciamento de recursos e representação de contexto em baixo nível, a arquitetura utiliza documentos XML.

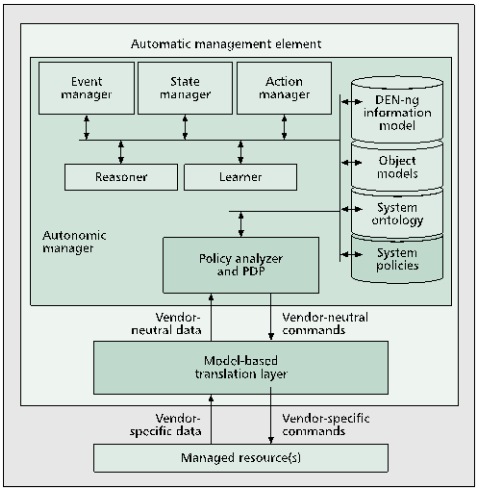


Figura 21: Estrutura da arquitetura FOCALE [47]

CASUP [43] foi proposta como uma arquitetura de personalização de ambientes baseando-se em informações de contexto e no perfil dos usuários cadastrados. O principal objetivo da arquitetura é realizar a personalização baseada em informações históricas, gerenciadas pela própria arquitetura.

A arquitetura é dividida nas seguintes camadas: **(i)** Camada de coleta de contexto: modela as informações coletadas por sensores. Estas informações são representadas em RDF. **(ii)** Camada de gerenciamento de contexto: Define um conjunto de regras em SWRL para o gerenciamento de informações de contexto representadas em RDF. Nesta camada são definidas regras que representam ações cotidianas. **(iii)** Camada de gerenciamento de preferências do usuário: Nesta camada é gerenciado o histórico de ações realizadas pelos usuários do sistema.

O número de ações realizadas interfere na personalização do sistema. A arquitetura utiliza árvores de decisão para a inferência de personalização baseada no histórico de ações (Figura 22) **(iv)** Camada de aplicação: Fornece APIs para a programação de aplicações para que estas utilizem as informações de contexto gerenciadas pela arquitetura.

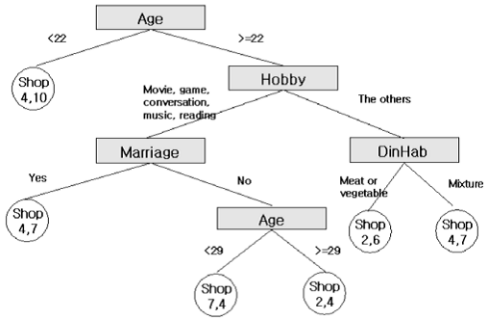


Figura 22: Árvore de decisão referente a ação *Jantar* [43]

A arquitetura ***EXEHDA-UC*** (*Execution Environment for Highly Distributed Applications – Ubiquitous Computing*) [53] é uma extensão da arquitetura EXEHDA. Ela foi definida com servidores em dois níveis: (i) Servidores de borda: são servidores que gerenciam a coleta de informações de contexto e as transformam em informações de alto nível, e (ii) Servidores de contexto: implementam a lógica de controle do contexto. Para a comunicação entre os servidores, foi utilizada a linguagem XML-RPC (*eXtensible Markup Language foi Remote Procedure Calls*). As informações de contexto são armazenadas em um banco de dados relacional.

O ***MultiS*** [30] é um servidor de contexto que gerencia a camada de reconhecimento de contexto. Este servidor utiliza a linguagem CD-XML (*Context Data-XML*) para descrever contextos em um nível acima das definições coletadas de sensores.

MoReCon [28] é uma arquitetura de gerenciamento de contexto para dispositivos móveis baseada em web services que utilizam o padrão REST. Desta forma, cada fonte de contexto é tratada como um recurso, e compartilhada via webservice. Assim, os autores garantem uma forma facilitada de acesso ao contexto, já que para acessar um determinado recurso os desenvolvedores precisam apenas acessar uma URL. Um exemplo com solicitação e atualização de recursos na MoReCon é apresentado na Figura 23.

GET	/location
POST	/location <geo> <lat>48.69096</lat><lon>9.14062</lon> </geo>
PUT	/location <geo> <lat>48.69096</lat><lon>9.14062</lon> </geo>
DELETE	/location

Figura 23: Exemplo de operações realizadas sobre o recurso localização na arquitetura MoReCon [28]

Utilizando os comparativos realizados por Perera et al [61], Makris et al [54] e Hong et al [43], montamos uma tabela que sumariza as principais arquiteturas de gerenciamento de contextos, com suas principais características e requisitos atendidos. As características analisadas sobre as arquiteturas são:

- **Modelagem (*a*)**: A modelagem de informações de contexto (Seção 2.2) pode utilizar diversas abordagens. As mais comuns são: Chave-Valor **(*C-V*)**, Esquemas de marcação **(*M*)**, Lógica Fuzzy **(*F*)**, Orientação a objetos **(*Oo*)**, Ontologias **(*On*)** e Diagramas gráficos **(*G*)**.

- **Inferência (*b*)**: As inferências (raciocínios) em contexto são extremamente importantes em arquiteturas baseadas em contexto, pois são elas que definem uma série de fatores de adaptação e personalização de recursos. Além disso, permitem com que sejam realizadas abstrações de contexto, para a identificação de situações no ambiente.

A inferência de contextos possui ainda dois requisitos importantes em sistemas baseados em contexto: (i) tratamento da imperfeição da informação bruta proveniente de sensores (informações desconhecidas, ambíguas, imprecisas ou erradas), e (ii) tratamento de incertezas.

A inferência pode ser realizada de diversas formas. As mais utilizadas são: Aprendizado Supervisionado (redes neurais, redes Bayesianas, Árvores de decisão) **(*AS*)**, Aprendizado Não Supervisionado (Clusterização) **(*ANS*)**, Regras Lógicas **(*R*)**, Lógica Fuzzy **(*LF*)**, Lógica de primeira ordem baseada em ontologias **(*LbO*)**, e Lógica Probabilística (Naive Bayes, Modelo oculto de Markov) **(*LP*)** [61][54][43].

- **Persistência e armazenamento de histórico** (*c*): A persistência de informações do ambiente é muito importante para sistemas ubíquos. Isto se deve principalmente ao fato de que é necessária a utilização de analises de informações históricas de atividades no ambiente para aumentar a eficiência na adaptação e personalização de sistemas. Os trabalhos analisados podem não armazenar o histórico de contexto **(-)**, armazenar e recuperar o histórico de contexto, porém sem a definição de qual tecnologia e modelagem utilizadas **(~)**, ou armazenar e recuperar informações de contexto, e informar quais tecnologias são utilizadas **(✓)**.

- **Processamento em tempo real** (*d*): O processamento de informações de contexto em tempo real permite com que sistemas baseados em contexto consigam agir no tempo necessário para modificar o ambiente. Devido a complexidade de algumas formas de modelagem e inferência, muitos sistemas utilizam formas híbridas de modelagem e raciocínio para permitir que pelo menos algumas partes de contexto possam ser analisadas em tempo real, enquanto inferências mais complexas são processadas em *background*.

- **Formas de integração entre serviços** (*e*): Sistemas sensíveis ao contexto devem se integrar ao ambiente e consequentemente, a sensores presentes nestes ambientes. Esta integração também deve ser feita entre módulos do próprio sistema sensível ao contexto, pois funções como a persistência, recuperação e inferência de contexto devem ser escaláveis. As principais formas de integração entre serviços são: Padrão SOAP **(*S*)**, padrão REST **(*R*)**, e *Remote Procedure Calls* **(*RPC*)**.

- **Gerenciamento de Conhecimento** (*f*): O gerenciamento de conhecimento é necessário em sistemas sensíveis ao contexto em diversos níveis, pois estes sistemas

precisam realizar operações sobre diversos tipos de informação. Informações sobre usuários, domínios e ambientes podem ser necessários para a realização de tarefas. Para a representação e gerenciamento de conhecimento, geralmente são utilizadas ontologias e motores de inferência, que em conjunto com regras lógicas definem as inferências feitas sob o conhecimento representado.

A Tabela 7 apresenta a sumarização das principais características de arquiteturas de gerenciamento de contextos.

Tabela 7: Comparativo entre arquiteturas de gerenciamento de contexto

Arquitetura/ Característica	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
a	C-V	M	M	On	On	C-V, On, Oo	G, On	M	M	O	On, M	M	M	M
b	✓	R	AS, LP	R, LbO	R, LbO	R, LbO	O	✓	R	LbO	R, AS	-	-	-
c	✓	-	✓	✓	✓	✓	✓	-	✓	-	✓	✓	-	-
d	-	-	-	-	-	-	-	-	✓	-	-	✓	✓	✓
e	✓	RPC	RPC	S	RPC	RPC	S	HTTP	S	S	✓	✓	✓	R
f	-	-	-	✓	✓	✓	✓	-	-	-	✓	-	-	-

1: ***Context Toolkit*** [27]; **2:** ***Aura*** [35]; **3:** ***Gaia*** [70]; **4:** ***CoBrA*** [20]; **5:** ***CROCO*** [62]; **6:** ***Hydra*** [8]; **7:** ***Feel@Home*** [37]; **8:** ***Octopus*** [31]; **9:** ***C-CAST*** [65]; **10:** ***FOCALE*** [47]; **11:** ***CASUP*** [43]; **12:** ***EXEHDA-UC*** [53]; **13:** ***MultiS*** [30]; **14:** ***MoReCon*** [28].

Observando a sumarização de arquiteturas de gerenciamento, podemos constatar algumas características comuns entre as arquiteturas:

- A utilização de ontologias impacta diretamente na forma de construção de arquiteturas de gerenciamento de contexto. As arquiteturas que utilizam ontologias como modelo de representação de contexto realizam o gerenciamento do conhecimento relacionado ao contexto e ao domínio, além de utilizarem regras de inferência baseadas em ontologias para a realização de inferências. Porém, a utilização de ontologias prejudica o processamento de contextos em tempo real, pois os motores de inferência associados utilizam muitos recursos computacionais. Além disso, comumente tem-se utilizado bancos de dados relacionais em conjunto com frameworks de persistência de ontologias. Isto faz com que sejam realizadas constantes conversões de formatos, prejudicando o desempenho e a representatividade das informações;

- A distribuição de serviços entre componentes das arquiteturas e outras aplicações na maioria das vezes é feita com os padrões SOAP e RPC. Estes padrões demandam maior complexidade de gerenciamento. O padrão REST, que garante a interoperabilidade entre linguagens e arquiteturas é utilizado em apenas um projeto, o qual não utiliza ontologias para a representação e processamento de contextos;

O principal foco das arquiteturas é gerenciar informações de contexto para permitir a adaptação de execução das aplicações. Porém, em nenhum dos trabalhos pesquisados, encontramos referencias de mecanismos para a facilitação da utilização de contexto para a

realização em consultas em bancos de dados que contém as informações de domínio das aplicações pervasivas.

5 Estudo de Caso

Artigos como os de Mark Weiser [86], que definiu cenários para a utilização de conceitos da computação ubíqua, e o de Tim Berners-Lee et al [11], que definiu um conjunto de conceitos de web semântica, nortearam o desenvolvimento de aplicações pervasiva e ubíquas, e tem servido de base para a definição de arquiteturas que integrem domínios, contextos e situações.

Nesta Seção, definimos um estudo de caso fictício que busca representar uma conjunto de situações em um ambiente próximo ao descrito por Mark Weiser. Através deste estudo de caso, elencamos um conjunto de requisitos importantes para a definição de arquiteturas de gerenciamento de contexto e de aplicações pervasiva. Através dos requisitos levantados, realizamos um comparativo entre as arquiteturas e soluções para o gerenciamento de contexto para analisar se estas soluções atendem aos requisitos levantados. Desta forma, foi possível identificar oportunidades de pesquisa na área.

O estudo de caso proposto é baseado na união de dois domínios de aplicação: Ambientes Inteligentes para assistência a saúde (*Ambient Assisted Living* – AAL), e sistemas hospitalares ubíquos (*HealthCare Systems*).

O paciente idoso João (70 anos de idade, com histórico de doenças cardiovasculares) está em sua casa. Esta casa possui um conjunto de sensores que coletam dados e os transmitem para uma arquitetura que monitora o ambiente, utilizando informações de contexto para adaptação na apresentação de informações aos usuários e na adaptação de ações (feitas por outros sistemas ou através de atuadores).

Em um determinado momento, João sofre uma queda. A arquitetura de monitoramento da residência notifica um cuidador através do celular, e como João está na sala, emite um alerta na televisão avisando que já fez o chamado. Como João demora para levantar, e o cuidador não enviou uma confirmação de que está indo para casa, a arquitetura chama o serviço de saúde contratado por João.

No momento em que chama o serviço de saúde, a arquitetura realiza uma consulta do histórico de ações recentes de João e de contextos (batimentos cardíacos e temperatura) nos momentos que que as ações foram feitas. O resultado desta consulta é informado ao sistema de gerenciamento do hospital, que anexa as informações a ficha hospitalar de João (juntamente com outras informações médicas sigilosas). As informações de ações e contextos, além de um resumo da ficha médica são transmitidos pelo sistema ubíquo do hospital a uma ambulância, que realiza o atendimento a João. Quando a ambulância chega a casa de João, a arquitetura de monitoramento confirma a chegada do socorro a arquitetura de gerenciamento do hospital.

A partir desde cenário, podemos definir um conjunto de requisitos desejáveis para as arquiteturas de gerenciamento de contextos:

- A modelagem de informações deve permitir a representação de situações (requer poder de expressividade), e deve permitir a modelagem e controle de fluxos contínuos de dados de sensores (no cenário, os sensores de temperatura e batimentos cardíacos mandam informações contínuas de monitoramento);

- Há a necessidade de existência de um meio de comunicação interoperável, independente de plataforma, para permitir a troca de informações de contexto entre arquiteturas e entre módulos de uma mesma arquitetura (considerando que sensores, atuadores e módulos são heterogêneos e desenvolvidos utilizando plataformas distintas);

- Informações semânticas sobre os contextos e situações ocorridas no ambiente precisam ser trocadas entre arquiteturas de diferentes domínios, com diferentes sistemas gerenciadores de bancos de dados (arquitetura residencial utiliza um banco de dados baseado em documentos e a arquitetura hospitalar utiliza um SGBD relacional para armazenar o histórico de informações do paciente.

Contextos e dados armazenados devem ser recuperados com base no contexto atual do ambiente. No cenário proposto, existe a necessidade de consultar apenas o histórico recente, de batimentos cardíacos e temperaturas da pessoa *João* nas ações que ele realizou recentemente.

6 Conclusões e Oportunidades de Pesquisa

As áreas de sensibilidade ao contexto e a situação estão evoluindo constantemente em diversos aspectos, principalmente no que se refere ao tratamento de informações de contexto vindas do ambiente e a utilização destas informações em sistemas ubíquos.

Baseado nos estudos realizados sobre CA e SWA, e na definição de um estudo de caso que envolve sistemas ubíquos de diferentes domínios, elencamos algumas oportunidades de pesquisa na área.

Modelagem:

- Contexto existe independentemente de interações do usuário com sistemas. Portanto, além das operações realizadas na detecção de eventos, operações devem ser feitas mesmo sem a interação do usuário, através da leitura de fluxos de informação (por exemplo na leitura de sensores em tempo real), ou de forma proativa (através da análise do histórico de informações de contexto e de ações previamente tomadas [54] [80];

- Ontologia é o modelo com maior poder de expressividade para a representação de contextos. O padrão mais utilizado para a representação de ontologias de contexto é o OWL-DL. Recentemente, a OWL 2 foi proposta como padrão de representação de ontologias, oferecendo um conjunto de melhorias em relação a eficiência e a expressividade. Porém, não foi possível localizar ontologias de base para representação de contexto em OWL 2. Além disso, há uma lacuna na utilização de ontologias em relação a mobilidade (suporte a movimentação de conceitos e processamento em dispositivos móveis), aplicabilidade (ligação com sistemas tradicionais) e facilidade de implementação (ligação com diagramas visuais de modelagem de contextos) [79] [12] [54];

- A ontologia *OWL-Time* representa os relacionamentos temporais definidos na álgebra de Allen. Estas definições, em conjunto com a definição de contextos e a utilização de bancos de dados que suportem ontologias pode permitir com que inferências sejam realizadas para a análise do histórico de situações ocorridas em um ambiente inteligente.

Distribuição:

- Ainda faltam iniciativas de padronização para a troca de informações de contexto. Esta questão é essencial para ecossistemas baseados em contexto e interfere diretamente em questões de escalabilidade e interoperabilidade [54][12];

- A maioria das arquiteturas de gerenciamento de contexto tratam de domínios isolados, porém não fornecem ferramentas para o compartilhamento de informações entre domínios. O JSON-LD [52][50] é um padrão de representação de *linked data* em JSON, para a utilização com *web services* REST. A extensão do JSON-LD para suportar os construtores de OWL-DL podem manter a expressividade de contexto de OWL-DL e fornecer meios de integração entre informações semânticas de diferentes domínios.

Persistência e Recuperação:

- Arquiteturas de gerenciamento de contexto estão diretamente relacionadas a ligação com sensores e dispositivos móveis. A capacidade de armazenamento e processamento de dispositivos móveis podem limitar a ação da arquitetura. Além disso, realizar todas as operações sobre o contexto em apenas um lugar (arquitetura cliente-servidor) não é recomendado, pois o ambiente é heterogêneo e distribuído [22][77][51].

- A recuperação de informação em bancos de dados de acordo com informações de contexto ainda é um desafio. Principalmente devido a heterogeneidade de sensores que interferem diretamente no contexto de alto nível [54];

- Análise de histórico está diretamente relacionada com a capacidade de sistemas ubíquos de persistência e recuperação de contexto. Se a modelagem é feita utilizando ontologias, geralmente são utilizadas as linguagens SWRL (para a realização de inferências) e SPARQL (para a realização de consultas em RDF). Porém, estas linguagens não oferecem mecanismos de monitoramento de fluxos de informação em tempo real, e possuem poucos recursos se comparadas as linguagens baseadas em SQL para consultas em relação a tempo e espaço [38][54];

- A eficiência na inferência e consulta em bases de dados que integram ontologias ainda são um problema para sistemas ubíquos. Além disso, estas abordagens não atendem aos requisitos de mobilidade necessários para sistemas ubíquos [12][79][54];

- A metodologia CARVE [13] propôs a geração automática de visões em bancos relacionais baseada em contexto (representado com um árvore de conceitos). Não foi possível localizar trabalhos que façam a geração automática de visões baseadas em contextos modelados em ontologias. Isto permitiria a recuperação de informação de repositórios de modelos relacionais utilizando uma forma comum de representação de contexto (ontologias).

7 Referências

[1] 3rd Generation Partnership Project. **Generic User Profile – architecture. Technical specification of Technical Specification Group Services and System Aspects**, Version 6.3.0. 2004

[2] Ahlgren B., Ahlgren, P. A. Aranda, P. Chemouil, S. Oueslati, L. M. Correia, H. Karl, M. Sllner and A. Welin, "**Content, Connectivity, and Cloud: Ingredients for the Network of the Future**", IEEE Commun. Mag.,vol. 49, no. 7, pp. 62-70, 2011.

[3] Al Feel, H. T., & Khafagy, M. H. (2011). **OCSS: Ontology Cloud Storage System**. *2011 First International Symposium on Network Cloud Computing and Applications*, 9–13. doi:10.1109/NCCA.2011.9

[4] Alliance, Open Mobile. "**User agent profile**." *Approved version* 2 (2006).

[5] Anderson, K. M., Hansen, F. A., Bouvin, N. O., "**Templates and queries in contextual hypermedia**," *Proc. seventeenth Conf. Hypertext hypermedia - HYPERTEXT '06*, p. 99, 2006.

[6] Andrejev, A., & Risch, T. (2012). **Scientific SPARQL: Semantic Web Queries over Scientific Data**. *2012 IEEE 28th International Conference on Data Engineering Workshops*, 5–10. doi:10.1109/ICDEW.2012.67

[7] Attard, J., Scerri, S., Rivera, I., and Handschuh, S., "**Ontology-based situation recognition for context-aware systems**," *Proc. 9th Int. Conf. Semant. Syst. - I-SEMANTICS '13*, p. 113, 2013.

[8] Badii, A., Crouch, M., Lallah, C., "**A context-awareness framework for intelligent networked embedded systems**," in Advances in Human- Oriented and Personalized Mechanisms, Technologies and Services (CENTRIC), 2010 Third International Conference on, aug. 2010, pp. 105 –110. [Online]. http://dx.doi.org/10.1109/CENTRIC. 2010.29

[9] Batzios, A., Mitkas, P.A., (2009), "**db4OWL: An Alternative Approach to Organizing and Storing Semantic Data**," *IEEE Internet Computing*, vol. 13, no. 6, pp. 48-55, Nov./Dec. 2009.

[10] Baumgartner, Norbert, and Werner Retschitzegger. "**A survey of upper ontologies for situation awareness**." *In Proceedings of the 4th IASTED International Conference on Knowledge Sharing and Collaborative Engineering*. 2006.

[11] Berners-Lee, Tim, James Hendler, and Ora Lassila. "**The semantic web**." *Scientific american* 284.5 (2001): 28-37.

[12] Bettini, C., O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "**A survey of context modelling and reasoning techniques**," *Pervasive Mob. Comput.*, vol. 6, no. 2, pp. 161–180, Apr. 2010.

[13] Bolchini, C., Quintarelli, E., Tanca, L., "**CARVE: Context-aware automatic view definition over relational databases**," *Inf. Syst.*, vol. 38, no. 1, pp. 45–67, Mar. 2013.

[14] BORST, W. **Construction of Engineering Ontologies for Knowledge Sharing and Reuse**. PhD thesis, University of Twente, P.O. Box 217 - 7500 AE Enschede - The Netherlands, 1997.

[15] Brezillon, P., Araujo, R. M.. **Reinforcing shared context to improve collaboration**. Revue d Intelligence Artificielle, 19(3):537–556, 2005.

[16] Brown, P. J., Jones, G. J. F., "**Context-aware Retrieval: Exploring a New Environment for Information Retrieval and Information Filtering**," *Pers. Ubiquitous Comput.*, vol. 5, no. 4, pp. 253–263, Dec. 2001.

[17] Carroll, Jeremy J., et al. "**Jena: implementing the semantic web recommendations**." *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, 2004.

[18] Chalmers, D. **Contextual mediation to support ubiquitous computing**. PhD thesis, Department of Computing, Imperial College, 2002.

[19] Chen H. et al, "**SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications**," em MobiQuitous, 2004.

[20] Chen, H., Finin, T., Joshi, A., Kagal, L., Perich, F., Chakraborty, D., "**Intelligent agents meet the semantic web in smart spaces**," IEE Internet Computing, vol. 8, no. 6, pp. 69 – 79, nov.-dec. 2004. http://dx.doi.org/10.1109/MIC.2004.66

[21] Chen, H., T. Finin e A. Joshi, "**An Ontology for Context- Aware Pervasive Computing Environments**," Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, vol. 18, pp. 197--207, 2003.

[22] Chun, B. G., Maniatis, P., "**Dynamically Partitioning Applications Between Weak Devices and Clouds**", in Proc. 1st ACM Workshop on Mobile Cloud Computing and Services (MCS), San Francisco, USA, 2010.

[23] Clark, Persia, "**Pellet: Owl 2 reasoner for java**," Software, http://clarkparsia.com/pellet/

[24] De Maio, C., Fenza, G., Furno, D., Loia, V., "**f-SPARQL extension and application to support context recognition**," *2012 IEEE Int. Conf. Fuzzy Syst.*, pp. 1–8, Jun. 2012.

[25] De Virgilio, R., Orsi, G., Tanca, L., & Torlone, R. (2012). **NYAYA: A System Supporting the Uniform Management of Large Sets of Semantic Data**. *2012 IEEE 28th International Conference on Data Engineering*, 1309–1312. doi:10.1109/ICDE.2012.133

[26] Dentler, K., Muhleisen, H.; Dentler, K., "**Large-Scale Storage and Reasoning for Semantic Data Using Swarms**," *Computational Intelligence Magazine, IEEE* , vol.7, no.2, pp.32,44, May 2012 doi: 10.1109/MCI.2012.2188586.

[27] Dey A. K., Abowd G. D., Salber D., "**A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications**," Hum.-Comput. Interact., vol. 16, pp. 97–166, December 2001. http://dx.doi.org/10. 1207/S15327051HCI16234 02

[28] Dogdu, Erdogan, Onur Soyer. "**MoReCon: a mobile restful context-aware middleware**." *Proceedings of the 51st ACM Southeast Conference*. ACM, 2013.

[29] Endsley, Mica R. "**Design and evaluation for situation awareness enhancement**." *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 32. No. 2. SAGE Publications, 1988.

[30] Fehlberg, F. W., Rolim , C. O., Leithardt , V. R. Q., Geyer, C. F. R., L. C. Silva, and A. G. M. Rossetto, "**MultiS: A Context-Server for Pervasive Computing**," *Electron. Notes Theor. Comput. Sci.*, vol. 292, pp. 39–56, Mar. 2013.

[31] Firner, B., Moore, R. S., Howard, R., Martin, R. P., Zhang, Y., "**Poster: Smart buildings, sensor networks, and the internet of things**," in Proc. 9th ACM Conference on Embedded Networked Sensor Systems, ser. SenSys '11. New York, NY, USA: ACM, 2011, pp. 337–338. http://doi.acm.org/10.1145/2070942.2070978

[32] Frà, C., Valla, M., "**High Level Context Query Processing : An Experience Report**," no. d, pp. 421–426, 2011.

[33] Friedman-Hill, Ernest. ***JESS in Action***. Greenwich, CT: Manning, 2003.

[34] Gama, J., Gaber, M.M.. **Learning from data streams: processing techniques in sensor networks**. Springer-Verlag New York Inc, 2007.

[35] Garlan D., Siewiorek D., Smailagic A., Steenkiste P., "**Project aura: Toward distraction-free pervasive computing**," IEEE Pervasive Computing, vol. 1, no. 2, pp. 22–31, Apr. 2002. http://dx.doi.org/10.1109/MPRV.2002.1012334

[36] Gu, T., Pung, H.K., Zhang, D.Q., **A service-oriented middleware for building context-aware services**, Journal of Network and Computer Applications 28 (1) (2005) 1–18.

[37] Guo, B., Sun, L., Zhang, D., "**The architecture design of a cross-domain context management system**," in 8th IEEE International Conference on Pervasive Computing and Communications Workshops, 29 2010-april 2 2010, pp. 499 –504. http: //dx.doi.org/10.1109/PERCOMW.2010.5470618

[38] Haghighi, P. D., Zaslavsky, A., Krishnaswamy, S., "**An Evaluation of Query Languages for Context-Aware Computing**," *17th Int. Conf. Database Expert Syst. Appl.*, pp. 455–462, 2006.

[39] HELD, A., BUCHHOLZ, S., AND SCHILL, A. **Modeling of context information for pervasive computing applications**. In Proceedings of SCI 2002/ISAS 2002 (2002).

[40] Henricksen, K. and Indulska, J.. **Developing context-aware pervasive computing applications: Models and approach**. *Journal of Pervasive and Mobile Computing*, 2(1):37–64, Feb. 2006.

[41] Henricksen, K., "**A Framework for Context-Aware Pervasive Comput- ing Applications**", PhD thesis, School of Information Technology and Electrical Engineering, The University of Queensland, Australia, 2003.

[42] Henricksen, K., Indulska, J., and McFadden, T.. **Modeling context information with ORM**. In *OTM Federated Confer- ences Workshop on Object-Role Modeling (ORM)*, volume 3762 of *Lecture Notes in Computer Science*, pages 626–635. Springer, November 2005.

[43] Hong J., Suh E. H., Kim J., Kim S., "**Context-Aware System for Proactive Personalized Service based on Context History**", Elsevier Expert Systems with Applications Journal, vol. 36, no. 4, pp. 7448- 7457, 2009.

[44] Hong, Jong-yi, Eui-ho Suh, and Sung-Jin Kim. "**Context-aware systems: A literature review and classification**." *Expert Systems with Applications* 36.4 (2009a): 8509-8522.

[45] Hoyos, José Ramón, Jesús García-Molina, and Juan Antonio Botía. "**MLContext: A Context-Modeling Language for Context-Aware Systems**." *Electronic Communications of the EASST* 28 (2010).

[46] Huq, M. R., Tuyen, N. T. T., Lee, Y.-K., Jeong e S. Lee, B.-S., "**Modeling an Ontology for Managing Contexts in Smart Meeting Stapace**," em SWWS, 2007, pp. 96--102.

[47] Jennings B., Meer S. V., Balasubramaniam S., Botvich D., O'Foghlu M., Donnelly W., Strassner J., "**Towards Autonomic Man- agement of Communications Networks**", IEEE Commun. Mag. ,vol. 45, no. 10, pp. 112-121, 2007.

[48] Kayes, A. S. M., Jun Han, and Alan Colman. "**An ontology-based approach to context-aware access control for software services**." *Web Information Systems Engineering–WISE 2013*. Springer Berlin Heidelberg, 2013. 410-420.

[49] Knappmeyer, M., Kiani, S. L., Fra, C., Moltchanov, B., and Baker, N., "**ContextML: A light-weight context representation and context management schema**," *IEEE 5th Int. Symp. Wirel. Pervasive Comput. 2010*, pp. 367–372, 2010.

[50] Kobayashi, M. N. Ishii, S. Takahashi, Y. Mochizuki, A. Matsushima, and T. Toyoda, "**Semantic-JSON: a lightweight web service interface for Semantic Web contents integrating multiple life science databases**.," *Nucleic Acids Res.*, vol. 39, no. Web Server issue, pp. W533–40, Jul. 2011.

[51] Kumar, K. and Y. Lu, "**Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?**", IEEE Computer, vol. 43, no. 4, pp. 51-56, 2010.

[52] Lanthaler, M. and C. Gütl, "**On Using JSON-LD to Create Evolvable RESTful Services**," no. April, 2012.

[53] Lopes, João, et al. "**Towards a distributed architecture for context-aware mobile applications in UbiComp**." *Proceedings of the 19th Brazilian symposium on Multimedia and the web*. ACM, 2013.

[54] Makris, P., Member, S., and Skoutas, D. N., "**A Survey on Context-Aware Mobile and Wireless Networking : On Networking and Computing Environments ' Integration**," vol. 15, no. 1, pp. 362–386, 2013.

[55] Maran, V.; Machado, A. ; Saccol, D. B. ; Augustin, I. . **A Software Architecture to Provide Persistence and Retrieve of Context Data Based on Ontological Models**. In: IADIS International Conference on WWW/Internet, 2011, Rio de Janeiro, Brasil.

[56] Martins, H. and N. Silva, "**Characterization, Comparison and Systematization of Context Ontologies**," *2012 Sixth Int. Conf. Complex, Intelligent, Softw. Intensive Syst.*, pp. 983–988, Jul. 2012.

[57] Matheus, C., Kokar, M., Baclawski, K., Letkowski, J., Call, C., Hinman, M., Salerno, J., D. Boul- ware, **SAWA: An assistant for higher-level fusion and situation awareness**, Proc. of SPIE Conference on Multisensor, Multisource Information Fusion: Ar- chitectures, Algorithms, and Applications, Orlando, Florida, USA, 2005, 75-85.

[58] Most, G. K., Pasquier, J., Scott, C., "**Context-Aware Computing : A Guide for the Pervasive Computing Community**," 2004.

[59] Neumann, C. P., Fischer, T., & Lenz, R. (2010). **OXDBS – Extension of a native XML Database System with Validation by Consistency Checking of OWL-DL Ontologies**, 143–148.

[60] Pan, Feng, and Jerry R. Hobbs. "**Temporal Aggregates in OWL-Time**." *FLAIRS Conference*. Vol. 5. 2005.

[61] Perera, C., S. Member, A. Zaslavsky, and P. Christen, "**Context Aware Computing for The Internet of Things : A Survey**," pp. 1–41, 2013.

[62] Pietschmann, S., Mitschick, A., Winkler, R., Meissner K., "**Croco: Ontology-based, cross-application context management**," in Semantic Media Adaptation and Personalization, 2008. SMAP '08. Third International Workshop on, dec. 2008, pp. 88 –93. [Online]. http://dx.doi.org/10.1109/SMAP.2008.10

[63] Preuveneers, D., J. Van Den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, and K. Coninx, "**Towards an extensible context ontology for Ambient Intelligence**," 2004.

[64] Reddy S., Burke J., Estrin D., Housen M., Srivastava M., "**Using Mobile Phones to Determine Transportation Modes**", ACM Trans. Sensor Networks, vol. 6, no. 2, pp. 1-27, 2010

[65] Reetz, E. S., Tonjes, R., Baker, N., "**Towards global smart spaces: Merge wireless sensor networks into context-aware systems**," in Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on, may 2010, pp. 337 –342. [Online]. http://dx.doi.org/10.1109/ISWPC.2010.5483728

[66] Reichle, R., Wagner, M., Khan, M. U., Geihs, K., Valla, M., Fra, C., Paspallis, N., Papadopoulos, G. a., "**A Context Query Language for Pervasive Computing Environments**," *2008 Sixth Annu. IEEE Int. Conf. Pervasive Comput. Commun.*, pp. 434–440, Mar. 2008.

[67] Riva, O., "**Contory: A Middleware for the Provisioning of Context Information on Smart Phones**", in proc. Middleware, pp. 219-239, 2006.

[68] Robin, Alexandre, and Michael E. Botts. "**Creation of Specific SensorML Process Models**." *Earth System Science Center-NSSTC, University of Alabama in Huntsville (UAH), HUNTSVILLE, AL* 35899 (2006).

[69] Robinson, R., Henricksen, K., Indulska, J., "**XCML: A Runtime Representation for the Context Modelling Language**," *Fifth Annu. IEEE Int. Conf. Pervasive Comput. Commun. Work.*, pp. 20–26, Mar. 2007.

[70] Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R. H., Nahrstedt, K., "**A middleware infrastructure for active spaces**," IEEE Pervasive Computing, vol. 1, no. 4, pp. 74–83, Oct. 2002. http://dx.doi.org/10.1109/MPRV.2002.1158281

[71] Rukzio, E., G. N. Prezerakos, G. Cortese, E. Koutsoloukas, and S. Kapellaki, "**Context for Simplicity : A Basis for Context-aware Systems Based on the 3GPP Generic User Profil**e," pp. 166–169, 2005.

[72] Schilit B, Theimer M. **Disseminating active map information to mobile hosts**. IEEE Network 1994; 8: 22–32

[73] Schreiber, F. A., L. S. Member, R. Camplani, M. Fortunato, M. Marelli, "**PerLa : a Language and Middleware Architecture for Data Management and Integration in Pervasive Information Systems**," vol. 38, no. 2, pp. 2–22, 2012.

[74] Shan, Y., Yu, J., & Chen, D. (2012). **Implementation Method of the Semantic Data to be Stored into the Relational Database**. *2012 International Conference on Computer Science and Service System*, 2075–2078. doi:10.1109/CSSS.2012.516

[75] Sheng, Q. Z., Benatallah, B., "**ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development**," *Int. Conf. Mob. Bus.*, pp. 206–212, 2005.

[76] Shoaib, M., & Basharat, A. (2010). **ERMOS: An Efficient Relational Mapping for Ontology Storage**. *2010 IEEE International Conference on Advanced Management Science(ICAMS 2010)*, 399–403. doi:10.1109/ICAMS.2010.5553141

[77] Simoens, P., De Turck, F., Dhoedt, B., Demeester, P., "**Remote Display Solutions for Mobile Cloud Computing**", IEEE Computer, vol. 44, no. 8, pp. 46-53, 2011.

[78] Sterritt R., Mulvenna M. and Lawrynowicz A., "**Dynamic and Con- textualised Behavioural Knowledge in Autonomic Communications**", Springer Verlag Berlin Heidelberg, LCNS 3457, pp. 217-228, 2005.

[79] Strang, Thomas, and Claudia Linnhoff-Popien. "A context modeling survey." *Workshop Proceedings*. 2004.

[80] Strassner, J. and O'Sullivan, D., "**Knowledge Management for Context- Aware Policy-based Ubiquitous Computing Systems**", in Proc. 6th International Workshop on Managing Ubiquitous Communications and Services, Spain, 2009, pp. 67-76.

[81] Ting, S. L., et al. "**RACER: Rule-Associated CasE-based Reasoning for supporting General Practitioners in prescription making**." *Expert Systems with Applications* 37.12 (2010): 8079-8089.

[82] Tsarkov, D., "**Fact++**," Software, 2007, http://owl.man.ac.uk/factplusplus/

[83] W3C. **Composite Capabilities / Preferences Profile** (CC/PP). http://www.w3.org/Mobile/CCPP.

[84] Wang X. H., Gu T., Zhang D. Q. e Pung H. K., "**Ontology Based Context Modeling and Reasoning using OWL**," em International Conference on Pervasive Computing and Communication, 2004, pp. 18--22

[85] 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. doi:10.1109/CSO.2012.139

[86] Weiser, Mark. "**The computer for the 21st century**." *Scientific american* 265.3 (1991): 94-104.

[87] Yau, Stephen S., Liu, Junwei, **Hierarchical situa- tion modeling and reasoning for pervasive computing**, Proc. of the 3rdWorkshop on Software Technolo- gies for Future Embedded and Ubiquitous Systems, Gyeongju, Korea, 2006, 5-10.

[88] Yılmaz, Ö., Erdur, R. C., "**iConAwa – An intelligent context-aware system**," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 2907–2918, Feb. 2012.

[89] Zhang, H., Wang, Z., Gao, Z., & Li, W. (2009). **Design and Implementation of Mapping Rules from OWL to Relational Database**. *2009 WRI World Congress on Computer Science and Information Engineering*, 71–75. doi:10.1109/CSIE.2009.290

[90] Zimmermann, Andreas, Andreas Lorenz, and Reinhard Oppermann. "**An operational definition of context**." *Modeling and using context*. Springer Berlin Heidelberg, 2007. 558-571.

[91] Calvo, Rafael A., and Sidney D'Mello. "**Affect detection: An interdisciplinary review of models, methods, and their applications**." *Affective Computing, IEEE Transactions on* 1.1 (2010): 18-37.

[92] Luo, Jia, ed. **Affective computing and intelligent interaction**. Vol. 137. Springer, 2012.

[93] Strang, Thomas, Claudia Linnhoff-Popien, and Korbinian Frank. "**CoOL: A context ontology language to enable contextual interoperability**." *Distributed applications and interoperable systems*. Springer Berlin Heidelberg, 2003.

[94] Motik, Boris, et al. "**OWL 2 web ontology language: Structural specification and functional-style syntax**." *W3C recommendation* 27 (2009): 17.

[95] Bellavista, Paolo, et al. "**A survey of context data distribution for mobile ubiquitous systems**." *ACM Computing Surveys (CSUR)* 44.4 (2012): 24.

[96] Ye, Juan, Simon Dobson, and Susan McKeever. "**Situation identification techniques in pervasive computing: A review**." *Pervasive and Mobile Computing* 8.1 (2012): 36-66.

[97] C. Chi and C. Ding, "**Context Query in Information Retrieval Kwok-Yan Lam**," pp. 2–7, 2002.