Managing Distributed and Heterogeneous Context for Ambient Intelligence

José Viterbo[1], Markus Endler1, Karin Breitman1, Laurent Mazuel[2], Yasmine Charif[2], Nicolas Sabouret[2], Amal El Fallah Seghrouchni[2] and Jean-Pierre Briot[1,2]

[1] Departamento de Informática
Pontifícia Universidade Católica (PUC-RJ)
22453-900 Rio de Janeiro, Brazil
[2] Laboratoire d'Informatique de Paris 6 (LIP6)
Université Paris 6 - CNRS
75015 Paris, France

1 Introduction

The *anywhere/any time* paradigm is becoming the new challenge to the conception, design, and release of the next generation of information systems. New technologies, like Wi-Fi networks and 3rd generation mobile phones, are offering the infrastructure to conceive information systems as ubiquitous, that is, systems that are accessible from anywhere, at any time, and with (almost) any electronic device. However, the use of such ubiquitous access to information systems require new conceptualizations, models, methodologies, and support technologies to fully explore its potential.

In this context, mobility introduces new accessibility scenarios and increases complexity. New issues, such as how to enable users to retain their ability to cooperate while located in different workplaces, the role of context and location in determining cooperation, the support for ad-hoc cooperation in situations where the fixed network infrastructure is absent or cannot be used are beginning to arise. The approaches and technologies for supporting these new ways of working are still under investigation. Nevertheless, a particularly interesting trend is exploring the Ambient Intelligence paradigm, a multidisciplinary approach that aims at the integration of innovative technologies that support user activities through specific services of the environment, which are provisioned with minimal user intervention. Essentially, an Ambient Intelligence system should be aware of the presence of a person, perceive the needs of this person and be able to adapt to the needs of the users in a relaxed and unobtrusive manner [20].

Ambient Intelligence (in the following, abbreviated as AmI) requires new environments for software development and deployment, where large quantities of different devices and sensors need to be integrated, building a programmable and auto-configurable infrastructure. Several projects, e.g. Gaia, CoBrA, CHIL, etc., have developed prototypes of such environments, but usually with focus only

at specific use cases, user tasks or application domains. Hence, most researches have come up with pragmatic, problem-specific solutions, which are difficult to generalize and port to other applications. However, we believe that in a few years, nearly every public and private space will be equipped with sensors and smart appliances that are able to automatically adapt to the preferences and demands of the local user(s) and by such provide special context-specific services to them. Such systems will be open, i.e. these spaces will potentially serve any user with a communication device (e.g., a smart-phone with powerful computing and multimedia capabilities), which will be the unique digital interface of the user with the ambient services and with the devices of other users. Thus, openness entails that both software agents responsible for user devices (e.g., agents for assisting the user), and agents responsible for smart spaces (e.g. agents that control the devices of a room according to its current use), must be prepared to interact with an *a priori* unknown set of other software entities.

In this chapter, we present existing technologies and current proposals toward the integration of heterogeneous entities within an Ambient Intelligence system. Practical realization of applications for AmI poses several challenges to software developers, many of them related to heterogeneity, dynamism (i.e. mobility) and decentralization. We make no claim of a complete or exclusive treatment of the subject. In fact, there are also several other related challenges [55], for example, identification of user intent, knowledge acquisition, negotiation, etc. But since these issues are a complex subject on their own right, in this paper we will discuss only challenges related to context reasoning, distribution, interoperability and heterogeneity issues.

Context reasoning for AmI is very complex due to the dynamic, imprecise and ambiguous nature of context data, the need to process large volumes of data, and the fact that reasoning needs to be performed in a decentralized, cooperative way among several entities of the system, e.g. entities representing spaces, devices or users. Decentralization takes the form of physical distribution of computing and sensor devices, of context providers and consumers, of entities responsible for reasoning and brokering, of applications and of users that may potentially engage into a spontaneous collaboration.

Besides, AmI spaces are intrinsically heterogeneous at several levels: At the infrastructure level, they include a wide range of appliances and gadgets with very specific data and control access protocols, and which are typically interconnected through different kinds of (wireless) networks with specific protocols and QoS parameters. Also the providers and the types of context information are usually very specific for each space and device, as well as their representations and models, making it difficult to achieve a common representation for different entities of the ambient context. A similar problem of heterogeneity can be identified at the level of services due to the very different kinds of ambient control functions provided, combined with the lack of standardized interfaces for service access. Finally, heterogeneity problems are found also at the level of knowledge representation and modeling, where systems may employ very different kinds of knowledge bases, descriptions and reasoning techniques. Hence, even if two ele-

ments are conscious of the same concrete fact, there is the problem of alignment of their knowledge representations.

In this chapter, we will focus mainly on decentralized context reasoning, and on semantic mediation, since we understand that these are some of the main challenges for enabling interactions among heterogeneous and context-aware entities in open and dynamic environments of Ambient Intelligence. In the following subsection we describe a simple scenario, which highlights several problems related to AmI, such as location-specific context-awareness, ontology based distributed reasoning, heterogeneous knowledge basis and semantic mediation.

1.1 Scenario

Silva is a Brazilian professor and researcher who works at PUC-Rio. He is visiting LIP6 with several other researchers. Their purpose is to have joint workshops related to a collaboration project. Silva carries with him his smart phone and his notebook, both executing the Campus middleware services dedicated to collecting and interpreting context information and for collective reasoning with other ambient services and applications. The devices also host some context-aware applications that support the platform's self-configuration to adapt to different situations, according to user's preferences and environment conditions.

When Silva arrives at LIP6, his Wi-Fi and GPS enabled smart phone (SMP-1) connects to the network, and using the current GPS data, queries a location service to find out that its user (Silva) is at LIP6. It then determines that this university is a partner institution of PUC-Rio; obtains the IP address of the Ambient management service at LIP6 and registers with it, indicating the user's identity and preferences.

The Ambient management service registers SMP-1 and determines that it belongs to Silva, a visiting professor from PUC-Rio. The system verifies that Silva is involved with the collaboration project and sets a workspace for him, communicating with a service running on Silva's notebook (NTB-1) to configure it to grant access to the proper network directories & services. This system also informs other project members at LIP6 about Silva's arrival.

A personal agenda application running on SMP-1 contacts the context infrastructure to be notified about the beginning of each event involving the whole project team, based on the project schedule and the location. Another application on SMP-1, the Configuration manager, requests to be notified whenever Silva is in a room in which an *activity* has started, so that it may set the smart phone to vibe-mode, and as soon as the activity ends, switch it back to the ring mode.

Notice that when this application interacts with the Ambient's local context provider, there could be a semantic mismatch between the terms "activity", used in the device's ontology, and the terms "meeting" or "class", used in the Ambient ontology. Due to this semantic mismatch, Silva's application would not get the expected response from the Ambient Service and would issue a request for semantic mapping from a mediation service, which would try to identify equivalence or subsumption among the concepts and adjust the local ontology

to reflect this new classification. Hence, we identify that the main requirements of AmI are context-specific reasoning capabilities (i.e. to enable the spaces and the interacting computing entities to "understand what is going on") and the ability to adapt services/behaviors to the current situation and user preferences. Being the AmI environment an open system, reasoning is inherently distributed.

Due to the intrinsic characteristics of Ambient Intelligence systems, ontological and distributed context-reasoning using multi-agent systems seems to be the most suitable development paradigm (i.e. each agent interacts with other agents to reinforce and complement its own knowledge about the context). However, the main problem with distributed reasoning is that heterogeneous knowledge bases and models have to be mapped (i.e. aligned, mediated), which leads to the problem of identifying and resolving semantic mismatch of knowledge representations.

1.2 Outline

The rest of this chapter is organized as follows: next section presents several fundamental concepts for AmI with which we deal in this chapter. Section 3 discusses the related work on context awareness for AmI. In Section 4 we review the main approaches to deal with the ontology alignment problem. Section 5 presents the Campus approach for dealing with context and semantic heterogeneity in AmI. Section 6 concludes the chapter.

2 Fundamental Concepts

2.1 Ambient Intelligence

Ambient Intelligence (AmI), i.e. "intelligent" pervasive computing, builds on three recent key technologies [2]: Ubiquitous Computing, Ubiquitous Communication and Intelligent User Interfaces. Ubiquitous Computing is the integration of microprocessors into everyday objects like furniture, clothing, white goods, toys, even paint. Ubiquitous Communication enables these objects to communicate with each other and the user by means of ad-hoc wireless networking. Intelligent User Interfaces enable the inhabitants of an AmI environment to control and interact with the environment in a natural (voice, gesture) and personalized way (preferences, context).

AmI aims at making use of those entities in order to provide users with an environment, which offers services when and if needed. One great challenge of such environments is how to adequately address the heterogeneity and dynamic nature of users, services and devices. Key issues of the development of AmI are context-awareness and reasoning and how to identify and activate the appropriate service within a continuously changing multitude of services [39]. The ultimate goal is to make the ambient services more *intelligent* and adaptive to the specific needs of their users.

2.2 Context Awareness

Context awareness is the ability of a system to sense the current environment and autonomously perform appropriate adaptations in regard to its optimal operation, general behavior and user interaction. When a user enters a new context, it is desirable that the applications on his devices be able to adapt to the new situation, and the environment be able to adapt its services to the presence of the new user.

There exist several definitions for context and context-awareness, but one of the most referenced one can be found in [19]: "*Any information which can be used to characterize the situation of an entity. An entity is a person, a place or an object which is considered relevant for the interaction between a user and an application, including the user and the application*". In an attempt to classify context, Chen and Kotz [16] identified four basic types of context: computational context (i.e. state of resources at the device and of the network), user context (i.e. persons, places and objects), physical context (e.g. luminosity, noise, temperature) and temporal context (e.g. hour, day, period of the year). Abowd et al [1] proposed the notions of *primary context* (localization, identity, activity and time) and of *secondary context*, where the latter one can be deduced from the former one and may be used for making adaptation decisions at a higher level of abstraction.

Conceptually, context provisioning can be organized in three layers [33]: data acquisition and distribution, interpretation and utilization. Before raw context data acquired from sensors and devices can be utilized, it must the interpreted and evaluated with respect to its accuracy, stability and reliability. The interpretation layer may also combine context data from different sources to enhance its reliability or completeness. For applications to be able to understand, describe and manage context-aware adaptations, it is necessary to have a context model, which can be defined at the application or the middleware layer. Strang and Linnhoff-Popien [62] identified and compared six types of context models: attribute-value pairs, schema-based models, graphic models, logic-based models, object-oriented models and ontology-based models. The author's main conclusion is that the object-oriented and the ontology-based models are the most complete and expressive ones, and hence are the most suited for modeling context for ubiquitous computing.

2.3 Ontology

Ontology has not only the advantage of enabling the reuse and sharing of common knowledge among several applications [58], but also of allowing the use of logic reasoning mechanisms to deduce high-level contextual information [68]. Therefore it has been widely adopted over other conceptual models, such as taxonomy, relational database schema and OO software models, for representing context information in ubiquitous systems.

A taxonomy is a set of terms arranged in a generalization-specialization (parent-child) hierarchy because they are much more expressive [34]. A controlled vocabulary simply lists a set of terms and definitions. A taxonomy may

or may not define attributes of these terms. A relational database schema defines a set of terms through classes, attributes and a limited set of relationships among those classes. An OO software model defines a set of concepts and terms through a hierarchy of classes and attributes and a broad set of binary relationships among classes. Constraints and other behavioral may be specified through methods on the classes (or objects).

An ontology can express all of the preceding relationships, models and diagrams as well as, n-ary relations, a rich set of constraints, rules relevant to usage or related processes and other differentiators including negation and disjunction [25]. Table 1 summarizes the benefits of the adoption of ontology.

- Ontologies are semantically richer, i.e. have greater expression power than taxonomies, entity relationships or OO models;
- Conceptual knowledge is maintained through complex and accurate representations above and beyond hierarchical approaches;
- Ontologies are formal — OWL DL ontologies map directly to Description Logic (a dialect of first order logics);
- Formal ontologies in the OWL DL standard can be verified/classified with the aid of Inference Mechanisms, e.g. RACER and FaCT:
  - consistency checks;
  - classification
  - new information discovery;
- OWL ontologies use a XML/RDF syntax that allows them to be automatically manipulated and understood by most resources on the Internet;
- Ontologies capture and represent finely granulated knowledge;
- Ontologies can be used to reduce ambiguity so as to provide a model over which information can be freely shared and acted upon by autonomic managers;
- Ontologies are modular, reusable and code independent — ontology driven applications are specified separately from the ontology itself. Changes to the ontology should not impact the code or vice versa;
- Ontologies can be combined with emerging rule languages, such as SWRL.

Table 1. Benefits of adopting formal ontology to model ambient knowledge in Campus.

2.4 Context Reasoning

Reasoning is necessary in context aware systems to deal with the intrinsic imperfection and uncertainty of context data, and also to infer secondary context data. Henricksen and Indulska [28] have characterized four kinds of context imperfectness: unknown, ambiguous, imprecise, and erroneous. The main tasks of reasoning are to detect possible errors, make estimates about missing values, determine the quality and validity of the context data, transform context data into meaningful information, and infer new, implicit context information, that

may be relevant for the applications. Reasoning is also fundamental for any kind of context-oriented decision-making, e.g. system adaptations according to user-provided or learned decision rules.

According to [46] reasoning for context-aware systems can be approached from four main perspectives: the low-level perspective, which includes basic tasks such as data pre-processing, data fusion, and context inference, usually performed by the sensors or the middleware, the application-oriented perspective, where the application can use a wide variety of reasoning methods to process the context data, the context monitoring perspective, where the main concern is a correct and efficient update of the knowledge base as the context changes, and, finally, model monitoring perspective, where the main task is to continuously evaluate and update learned context classifiers/interpreters and their models, also taking into account user feedback. Although Nurmi and Floren give an interesting perspective on context reasoning, we understand that instead of four perspectives, these are in fact complementary tasks, which should be present in every approach for reasoning in context-aware systems.

For context reasoning, several approaches have been adopted: ontological reasoning, rule-based reasoning, distributed reasoning and probabilistic reasoning [7]. Instead of presenting and comparing the general reasoning approaches, which are very well surveyed in Bikakis *et al* [7], in this paper, we will focus only on ontological and distributed reasoning approaches. On the one hand, ontologies offer high expressiveness and the possibility to develop a formal context model that can be shared, reused, extended to specific domains, and on the other hand, distributed reasoning is a direct requirement that arises from the open, dynamic and heterogeneous nature of AmI.

In the next section, we review the main approaches to deal with the contextual reasoning and ontological representations for AmI. Section 4 will focus on ontology alignment and semantic mapping between concepts, to deal with semantic heterogeneity in AmI. Section 5 will present our proposition to tackle both issues within the Campus framework.

3 Ontological representation and reasoning about context

In this section we survey several research work that deal with ontological representation and reasoning about context for Ambient Intelligence. We first present the main criteria used for comparison and a proposed taxonomy; then we present each work with respect to each criteria; and finally, we classify the systems according to our taxonomy and discuss their suitability for implementing Ambient Intelligent environments.

3.1 Evaluation Criteria and Taxonomy

There is much research work on middleware systems that support context modeling and ontological reasoning about context. However, as expected, each one is based on a different notion of context, uses ontologies in a different way, has

specific goals and approaches for context-specific reasoning and handling heterogeneity, and is targeted at specific applications or use scenarios. In this section, we will compare the works in regard to the following criteria:

1) Types of context. Which types of context information are collected, processed and distributed by the system (e.g. system context, location, physical context, user role, preferences, etc.). This information will give an idea of the framework's usefulness, scalability and practical feasibility.

2) Ontologies. Which ontologies are used, and for which purpose? What sorts of concepts and relationships are represented? Is the ontology extensible? How context instances are updated and persisted, etc.? This criterion assesses the system's expressiveness and flexibility.

3) Inferences/Reasoning technique. What kind of reasoning is supported? What sorts of higher-level context is inferred? Does the work consider uncertainty of the inferred context? This aspect determines the expressive power, reliability, completeness and preciseness of the systems reasoning, as well as its practical applicability.

4) Knowledge Management. Is the knowledge base static, or do most of the facts in the knowledge require continuous updates? Does the system handle decentralizedheterogeneous knowledge bases, and if so, do they handle evolving knowledge models (ontologies). If heterogeneity is supported, what is the basic mediation or semantic alignment technique employed and how powerful is it? The evaluation with regard to this aspect will give insight on how well the system is suited to deal with the inherently dynamic, decentralized and unpredictable nature of Ambient Intelligence.

5) Architecture. Is the system based on a centralized, fully decentralized, or hybrid architecture, with respect to the knowledge bases, the reasoning process and the mediation/brokerage support. By discussing this aspect, we have an idea on the system's scalability, reliability and of the implicit execution overhead related to the distributed interactions.

Although there are many possible means of classifying the context systems, we believe that the following aspects are the most relevant for assessing their suitability for developing, open and heterogeneous Ambient Intelligence environments. Hence, we will use them as the basis for our taxonomy.

1. Centralized *versus* decentralized knowledge base;
2. Static *versus* dynamic (or extensible) set of context providers;
3. Main goal of context reasoning: enhance reliability of context information, derive higher-level context facts, or both;
4. Means of handling heterogeneous knowledge bases, if any.

In the following subsections we summarize and analyse the most representative middleware systems with regard to the presented criteria, and in Subsection 3.10, classify each system according to the proposed taxonomy.

3.2 Gaia

Gaia provides a generic computational environment that integrates physical spaces and their ubiquitous computing devices into a programmable computing and communication system [52]. It is similar to traditional operating systems in that it manages the tasks common to all applications built for physical spaces [50]. Each space is self-contained, but may interact with other spaces. Gaia provides core services, including events, entity presence (devices, users and services), discovery and naming. By specifying well-defined interfaces to services, applications may be built in a generic way so that they are able to run in arbitrary active spaces. Gaia uses CORBA to enable distributed computing. Gaia is a mature project. The first prototypes were implemented in 2002 and several applications for active-classrooms have already been developed.

Types of Context. The Gaia Context Infrastructure allows applications to obtain a variety of contextual information. Various components, called Context Providers, obtain context from either sensors or other data sources. These include sensors that track people's locations, room conditions (for example, temperature and sound) and weather condition. Context Providers allow applications to query them for context information. Some Context Providers also have an event channel to asynchronously send context events. Thus, applications can either query a Provider or listen on the event channel to get context information.

Ontologies. Gaia's context model is based on first-order predicates. The name of the predicate indicates the type of context that is being described (e.g. location, temperature or time), and its typed arguments describe the properties of the context. For example, if the predicate is "location", the first argument has to be a person or object, the second argument has to be a preposition or a verb like "entering", "leaving", or "in" and the third argument must be a locationID. The structures of different context predicates are specified in an ontology. Each context type corresponds to a class in the ontology, which also defines the corresponding arguments of the predicate. Moreover, Gaia uses ontologies to describe various concepts of an Ubiquitous Computing Environment, such as kinds of applications, services, devices, users, data sources and other entities. They also define all terms used in the environment and the relationships between different terms. These ontologies are written in DAML+OIL.

Inferences/Reasoning techniques. Context Synthesizers are Gaia components that get sensed context data from various Context Providers, derive higher level or abstract context from these lower-level context data and provide these inferred contexts to applications. Whenever a Synthesizer deduces a change in the inferred context, it publishes the new information. Gaia adopts two basic inference approaches. Rule-based Synthesizers use pre-defined rules written in first order logic to infer different contexts. Each of the rules also has an associated priority, which is used to choose one rule when multiple rules are valid at the same time. However, if all the valid rules have the same priority, one of them

is picked at random. Alternatively, some Synthesizers may use machine learning techniques, such as Bayesian learning and reinforcement learning, to infer high-level contexts. Past context information is used to train the learner.

Knowledge Management. All the ontologies in Gaia are maintained by an Ontology Server. Entities contact the Ontology Server to get descriptions of other entities in the environment, information about context or definitions of various terms used in Gaia. The server also supports semantic queries, to get, for instance, the classification of individuals or subsumption of concepts. The Ontology Server also provides an interface for adding new concepts to existing ontologies. This allows new types of contexts to be introduced and used in the environment at any time. The Ontology Server ensures that any new definition is logically consistent with existing definitions. Since the ontologies clearly define the structure of contextual information, different agents can exchange different types of context information easily. For example, Context Providers and Context Synthesizers can get the structure of contexts that they provide, while Context Consumers query the Ontology Server for the structure of the requested context, and then frame appropriate queries to Context Providers to get the context information they need.

Architecture. The Gaia kernel consists of a Component Management Core that dynamically loads, unloads, transfers, creates, and removes any Gaia component or application. Each active space is self-contained but may interact with other spaces. For each space, Gaia manages its resources and services; provides location, context, and event services; and stores information about it. Gaia provides a set of basic services to be used by all applications. Among them, the Space Repository stores information about all software and hardware entities in the space and lets applications browse and retrieve an entity on the basis of specific attributes. The Space Repository learns about entities entering and leaving the active space through the Presence Service, which detects and maintains soft state information about applications, services, devices, and people in a active space. When the Presence Service detects that an entity is no longer available in an active space, it notifies the rest of the space that the entity left. In the context infrastructure, the Context Provider Lookup Service allows searches for different context providers. Providers advertise the set of contexts they provide in the form of a first order expression that describes the context provided. Applications can query the Lookup Service for a context provider that provides contextual information it needs.

3.3 CoBrA

Context Broker Architecture (CoBrA) is an infrastructure that supports agents, services and devices that interact in order to explore context information in active spaces [17, 18]. Its main component is an intelligent agent called *context broker*, which is responsible for providing a common model to represent context information, mediating the information exchanged between context providers

and resource constrained context consumers, and inferring higher-level context information not directly available from sensors [17]. In addition, the context broker is able of detecting and correcting inconsistent context data, and supports the enforcement of privacy policies defined by the users to control the sharing of their contextual information among other users. The proposed architecture is based on a central entity that was implemented as a FIPA-compliant agent using Jade.

Types of Context. CoBrA has a context-acquisition module, which is a set of library procedures for acquiring contextual information from sensors, agents and the Web. This library includes procedures for collecting information from Smart Tag sensors (location) and environment sensors (temperature, sound, luminosity, etc), but any other information can be added.

Ontologies. The base ontologies used for representing context information are the CoBrA Ontology (COBRA-ONT) and SOUPA. COBRA-ONT is a set of ontologies for agents to describe contextual information and to share context knowledge. It defines concepts for representing actions, agents, devices, meetings, time, and space. The SOUPA ontology, on the other hand, is a standard ontology for supporting pervasive and ubiquitous computing applications. It consists of vocabularies for expressing common concepts that are associated with person, agent, belief-desire-intention (BDI), action, policy, time, space and event, and also a set of vocabularies for supporting specialized domains of pervasive computing, such as smart spaces and peer-to-peer data management. The developer of a new system must design its specific ontology reusing some others that may be adequate.

Inference/Reasoning techniques. CoBrA's context reasoning is backed by the Jena rule engine, the Java Expert System Shell (JESS) and the Theorist system. The reasoning for interpreting context information uses two different rule-based systems. Jena rule-based reasoners are used for OWL ontology inferences and the JESS rule engine is used for interpreting context using domain specific rules. CoBrA supports also reasoning for maintaining a consistent context model by detecting and resolving inconsistent information, and the Theorist system is used for supporting the necessary logical inferences in that case. When a new context data is asserted into the knowledge base, the context broker first selects the type of context it attempts to infer (such as a person's location or a meeting's state). If such information is unknown, the broker decides whether it can infer this type of context using only ontology reasoning (Jena Rules). If logic inference is required, the context broker attempts to find all essential supporting facts by querying the ontology model and asserts them into the Jess engine. Before asserting the new inferred information into the knowledge base, ontology reasoners are used to infer whether the context described by the instant data is consistent with the model defined by the ontology. If not, a Theorist assumption-based reasoning is used for resolving inconsistent information.

Knowledge Management. The system provides a centralized (and homogeneous) model of context that all devices, services, and agents in the space must share. The knowledge of the context broker is represented as RDF statements and is stored in a persistent knowledge base. To acquire contextual information, all agents must send query messages to the context broker.

Architecture. CoBrA has a centralized architecture, where a single context broker agent should be deployed and all computing entities must be aware of this broker from the beginning. Usually, a single context broker is sufficient to support a small-scale smart space. However, being the main service provider in the space, the context broker may become the bottleneck of the system and a single point of failure. A team of context brokers can be deployed to overcome this problem, as well to improve system robustness through redundancy.

3.4 Semantic Space

Semantic Space [67, 68] is a context infrastructure developed to address three key issues. First, it aims to provide an explicit representation of the raw context data that is obtained from various sources in different formats. Furthermore, it provides means for the applications to selectively access a subset of context data through expressive context queries. Finally, it provides reasoning capabilities for inferring higher-level contexts. A prototype of the context infrastructure has been developed, and a prototype context-aware application was also implemented. The application, called SituAwarePhone, adapts mobile phones to changing situations while minimizing user distraction.

Types of Context. In Semantic Space, *context wrappers* obtain raw context information from various sources such as hardware sensors and software programs and transform them into context data. Some context wrappers work close to the hardware sensors deployed in the prototypical smart space, gathering information such as user's location, environmental temperature, noise, and light, status of doors (open or closed) of rooms, etc.. Software-based context include the activity of the user, based on the schedule information from Outlook Web Access; the status of different networked devices (such as voice over IP or mobile phones), the status (idle, busy, closed) of applications such as JBuilder, Microsoft Word, and RealPlayer from their CPU usage; and weather information obtained by periodically querying a weather web service.

Ontologies. Semantic Space uses the CONtext ONtology (CONON) for modeling context in pervasive computing environments [68]. Rather than completely modeling all sorts of context in different kinds of smart spaces, this ontology aims to be an extensible upper-level context ontology providing a set of basic concepts that are common to different environments. To characterize smart spaces, there are three classes of real-world objects (user, location, and computing entity) and one class of conceptual objects (activity), which together form the skeleton of a "contextual-rich environment". Consensus domain ontologies

such as friend-of-a-friend (FOAF), RCAL Calendar, and FIPA Device Ontology were also integrated into CONON to model users, activities, and device contexts, respectively.

Inference/Reasoning techniques. Two context reasoners are available, a description logic based reasoner and a first-order logic based situation reasoner, both implemented using Jena Semantic Web Toolkit to perform forward reasoning over the knowledge base. The description logic based reasoner was built to carry out ontology reasoning. The more flexible first-order logic based situation reasoner deduces a wide range of higher-level, conceptual context from relevant low-level context, such as user's activity. Semantic Space requires developers to write rules describing higher-level context information for each particular application based on its needs.

Knowledge Management. In each smart space resides a *Context Knowledge Base*, which provides persistent context knowledge storage. It stores the extended context ontology for a particular space and the context data provided by users or gathered from context wrappers. The *Context Aggregator*, is responsible for discovering context wrappers, gathering context data from them, and then asserting the gathered data into the context knowledge base. It updates the knowledge base whenever a context event occurs. The scope of contexts that the knowledge base manages may change depending on the availability of wrappers. When a context wrapper joins the smart space, the context aggregator adds the provided contexts to the knowledge base, and when the wrapper leaves, the aggregator deletes the contexts it supplied to avoid stale information.

Architecture. The architecture is centralized around a *Context Aggregator* and a *Context Knowledge Base*. Developers can add new wrappers to expand the scope of contexts in a smart space or remove existing wrappers when the contexts it provides are no longer needed.

3.5 CHIL

The middleware infrastructure developed in the CHIL (Computers in the Human Interaction Loop) Project [60] provides mechanisms for service access, context modeling, control of sensors and actuators, directory services for infrastructure elements and services, as well as fault tolerance mechanisms. In general, this middleware infrastructure allows developers to focus on the service logic, rather than on the details of context processing and utility services, also providing a framework with several components that can be reused across different ubiquitous computing services. Mechanisms for modeling composite contextual information and describing networks of situation states are also available. The middleware has been implemented as a distributed multi-agent system where the agents are augmented with fault tolerance capabilities using the agent's capacity to migrate between hosts.

Types of Context. The infrastructure can exploit numerous sensors for context acquisition, and new sensors can be plugged into the framework to provide information that may be used to compound derived contextual information or define situations that will trigger system's responses. Context information is obtained from sensors by software agents and made accessible to other agents of the system through the *Knowledge Base Agent*. Monitoring and control of sensors is performed through special *Proxy agents* that represent the sensors in the world of agents. Each proxy agents exposes a *universal virtualized interface* to the agent framework. A sensor specific driver is required to adapt the universal interface commands to the low-level capabilities of each particular sensor. This low-level driver is based on the control API offered by the sensor. Actually, three concrete proxy agents were implemented: one generic, one for microphones and one for cameras.

Ontologies. The CHIL ontology aims to establish a general-purpose core vocabulary for the various concepts comprising a multi-sensor smart space and the context-aware applications associated [47]. It was modularized to allow different parts to be used in different contexts and applications. Separated namespaces are used so that developers may safely introduce new concepts locally in their module's name-space without interfering with other modules. Assuming that other modules use similar concepts that should be merged, the core module may provide a merged version of the concept. To globally put together all the modules, the ontology consists of a main OWL file, which imports all modules. Developers interested only in a subset of modules can define a main OWL file of their own that imports only the modules of interest. The main component is the core module *chil-core*, which introduces concepts of perceivable entities such as, for example, *Person*, *MeetingRoom*, *Table* or *Whiteboard*, as well as perceivable roles of such entities, such as the *Location* of a *Person* or the *ActivityLevel* of a *MeetingRoom*.

Inference/Reasoning techniques. The approach adopted by CHIL to infer high-level contexts is based on the notion of *networks of situation states*. According to this approach a situation is considered as a state description of the environment expressed in terms of entities and their properties. Changes in individual or relative properties of specified entities correspond to events that signal a change in the situation. The concept of *role* serves as a variable' for the entities to which the relations are applied, thus allowing an equivalent set of situations to have the same representation. A role is played by an entity that can pass an acceptance test for the role, in which case, it is said that the entity can play or adopt the role for that situation. For example, in the scope of a meeting involving short presentations, at any instant, one person plays the role' of the presenter', while the other persons play the role of attendees'. Dynamically assigning a person to the role of presenter' makes it possible to select sensors to acquire images and sound of the current speaker. Detecting a change in some role allows the system to reconfigure the video and audio acquisition systems.

Knowledge Management. The knowledge base was developed as a server accessible both locally and remotely through a unique interface. The server remote interface is programming language independent, so that client components may be written in a variety of programming languages. The knowledge base server API is tailored to OWL.

Architecture. This architecture is centralized around some core agents, which are independent of the service and smart room installation. They provide the communication mechanism for the distributed entities of the system, control of the sensing infrastructure, and allow service providers to register their service logic into the framework. Besides, some agents that provide basic services such as the ability to track composite situations, the control of sensors, access to the knowledge base, are tightly coupled with the installed infrastructure of each smart room.

3.6 SAMOA

SAMOA framework [8] supports the creation of semantic context-aware social networks, which consist of logical abstractions that represent groups of mobile users who are in physical proximity and share common affinities, attitudes, and social interests. In particular, SAMOA lets mobile users create *roaming social networks* that, following user movements, at each instant reflect all nearby encounters of interest. Mobile users interested in creating social networks are called *managers*. They are responsible for defining the scope (i.e. radius) of discovery of their social network and the selection criteria. Other users located within the discovery boundaries are those *eligible* to become members of the *manager*'s social network. But only the users that are selected by the *manager* become *affiliated* with that social network.

Types of Context. To support the creation of social networks in ubiquitous environments, SAMOA relies on geographical context information, e.g. a user's location and reciprocal proximity, user attributes and social preferences, and place descriptions. Users' location and proximity are determined either by the network cell (or the WiFi access point) the user is currently attached to, or by the number of network hops between users in an ad-hoc network. The middleware provides graphic tools for specifying profiles of users and places.

Ontologies. SAMOA models and represents context data in terms of semantic metadata. Places and users are the entities in the system. They are associated with profiles describing their characteristics. A *place profile* has an identification and an activity parts. The former includes a unique identifier, a name and a description of the physical place, and the latter includes all of the social activities that characterize the place, and which sorts of information members located in that place are expected to share. The *user profile* consists of an identification and a preference parts. The identification part provides user naming information and describes user properties, such as age, gender and education, and the

preference part defines the activities the user is interested in and, for each of these activities, the user's specific preferences. Besides *place* and *user profiles*, managers also have a *discovery profile* associated with each place, defining which preferences *user profiles* must match to join the manager's social network at that place. Preferences in *discovery profile* include desired client attributes for each activity. While activities and preferences in the *place profile* and in the manager's *discovery profile* are represented as classes, activities and preferences in a *user profile* are defined as instances.

Inferences/Reasoning techniques. SAMOA exploits two semantic matching algorithms for analyzing profiles and inferring potential semantic compatibility among users. The first algorithm operates on *user* and *place profiles* to identify a first set of eligible members located within an area of interest around a place. Only those users whose profiles have activities that are semantically related to that of the *place profile* activities become *eligible members*. The second matching algorithm selects among the previously selected *eligible members* only those users whose attributes semantically match the preferences included in the manager's *discovery profile* for that particular place. Moreover, the matching algorithms perform also ontology reasoning to identify if the activity or preference in the *user profile* is an instance of a more generic activity or preference class, or an instance of a more specialized activity or preference class in the manager's *place* or *discovery profile*. SAMOA relies on the Pellet DL reasoner [59] for implementing both matching algorithms.

Knowledge Management. No centralized database is kept in SAMOA. Place and discovery profiles are maintained and analyzed separately. The user's mobile devices keep their own user profiles. Some users that may become managers of a social-network keep on their devices discovery profiles associated with each place. Stationary devices may keep place profiles for each place. The manager communicates only the place profile to co-located users, preserving the privacy of its discovery profile. Similarly, users return their user profiles only to managers that provided places with activities of interest. In addition, keeping place and discovery profiles separate lets SAMOA distribute the overhead of the social-network extraction among all users, since the semantic analysis of the *place profile* is performed on user's devices, and semantic matching between *discovery* and *user profiles* is performed on manager devices.

Architecture. The SAMOA middleware has totally distributed architecture organized in two logical layers: *the basic service layer* and the *social-network management layer*. The basic service layer provides facilities for naming, detection of co-located users, and device communication. In this layer, the locationproximity manager (L/PM) lets SAMOA entities advertise their online availability by periodical broadcasts of advertisement messages. L/PM senses incoming advertisements and builds a table of "discovered" co-located users. The social-network management layer includes facilities for semantic-based social network extraction and management. In this layer, the *place-dependent social-network manager*

(PSNM) creates and maintains a table that includes all members of the manager's social-network that are currently co-located with the manager. The *global social-network manager* (GSNM) keeps a record (in a dedicated table) of all place-dependent social networks previously formed at the visited places, i.e. the manager's global social network. In addition, the table stores the *place profile* and the *discovery profile* of the manager, which guided the selection of each member.

3.7 CAMUS

Context-Aware Middleware for URC (Ubiquitous Robotic Companion) System (CAMUS) is a context-aware infrastructure for the development and execution of a network-based intelligent robot system [36]. It was designed to overcome limitations of the ubiquity, context-awareness, and intelligence that existing mobile service robots have. CAMUS gathers context information from different sensors and delivers appropriate context information to different applications. Moreover, CAMUS provides context-aware autonomous service agents that are capable of adapting themselves to different situations.

Types of Context. In CAMUS, a *sensor framework* processes input data from various sources such as physical sensors, applications and user commands and transfers them to the *Context Manager* through an *Event System*. The Context Manager manages context information collected from the Sensor Framework. When context information in the environment is changed, the Context Manager transfers events to the Event System. The context represented includes the user context, environment context, and computing device context. User context includes user profile, user's task information, user preference, etc. Environment context includes hierarchical location information, time, etc. Computing device context includes information about available sensors and actuators.

Ontologies. The context model in CAMUS is represented as a four-layered space, where each layer has a different abstraction level. In the *common ontology layer* are modeled the ontology concepts that are commonly used in various applications. The common ontology provides the high-level knowledge description to context-aware applications. Generally, highly abstracted knowledge can be easily reused by various applications. The *domain ontology layer* comes below the *common ontology layer*. It provides the domain specific knowledge to context-aware applications. This layer is composed of the *infrastructure domain ontology* and a set of *specific domain ontologies* for the application. The *infrastructure domain ontology* is the schema of the context model that is represented and managed in the context-aware system. The *specific domain ontology* is about specific services, for example, a presentation service. The *domain ontology layer* provides the schema to the layer below, the *instance layer*, where instances of the ontology concepts are represented. Above the *common ontology layer* there is the *shared vocabulary layer*, where is defined a set of shared vocabulary (and their semantics) used in the *common ontology layer*.

Inferences/Reasoning techniques. CAMUS context reasoning engine include many different reasoners, which handle the facts present in the repository and produce higher-level contexts [26]. The reasoning service is used by some context mapping services and context aggregators. They invoke the reasoners through a fixed API, providing the reasoners with context data. All new inferred facts will be inserted into that context data for later queries. The use of a fixed interface for all kinds of reasoning engine makes it possible to add and handle different reasoners. Multiple reasoning mechanisms are available. Reasoners can infer high-level contexts using rules written in different types of logic like first order logic, temporal logic, description logic (DL), higher order logic, fuzzy logic, etc. Instead they can also use various machine learning techniques, such as Bayesian learning, neural networks, reinforcement learning, etc. The middleware defines wrappers for each reasoner type. Besides, a Racer server [27] provides ontology reasoning to infer subsumption relationships, instance relationships, and consistency of context knowledge base.

Knowledge Management. The application context model is stored in the CAMUS context storage through a Knowledge Base adaptor. Applications can refer and change application context through Jena APIs. Moreover, application context model can be updated and changed through Jena rule engine and OWL reasoner depending on application-specific inference rules and subsumption reasoning.

Architecture. CAMUS has a centralized architecture composed of three parts: Main Server, Service Agent Manager and Service Agents. The Main Server manages context information delivered from Service Agent Managers. It generates and disseminates appropriate events to applications according to the context changes. The Service Agent Manager provides the container where Service Agents are executed. A Service Agent is a software module that acts as a proxy to connect various external sensors and smart devices to CAMUS. It delivers information of sensors in environment to the Main Server, receives control commands from the Main Server, controls devices in the environment, and conducts applications. The entities in the system communicate using PLANET, a lightweight and fault-tolerant communication mechanism which also supports the disconnected operations and asynchronous operations.

3.8 OWL-SF

The distributed semantic service framework, OWL-SF [44], supports the design of ubiquitous context-aware systems considering both the distributed nature of context information and the heterogeneity of devices that provide services and deliver context. It uses OWL to represent high-level context information in a semantically well founded form. Devices, sensors and other environmental entities are encapsulated and connected to the upper context ontology using OMG's Super Distributed Objects technology [54] and communicate using the Representational State Transfer protocol [23]. Integrated reasoning facilities perform the automatic verification of the consistency of the provided service specifications

and the represented context information, so that the system can detect and rule out faulty service descriptions and can provide reliable situation interpretation. A prototype of the system has been implemented and tested.

Types of Context. OWL-SF uses Super Distributed Objects (SDOs) [54] to encapsulate context providers, which may be sensors, devices, user's interfaces (GUIs) or services.

Ontologies. Each SDO that encapsulates context providers and service-providing devices is an OWL-SDO. This OWL extension adds new methods to a standard SDO which allow accessing the current state of an object as an OWL description. Each functional entity implemented as OWL-SDO has to be described using its own ontology containing terminological knowledge that enables the automatic classification of the object into appropriate service categories. The state of an object stores context values and is represented by an instance of a class in the ontology.

Inferences/Reasoning techniques. Deduction servers (DSs) are specific OWL-SDO with an RDF inference mechanism and an OWL-DL reasoner. The rule-based reasoning process is provided by the RDF inference component and the deduced facts are used to trigger events to other SDOs and to process service calls. A subscription notification mechanism is used to monitor the SDO parameters to generate notifications whenever an observed parameter changes, triggering the deduction process to update the global ontology model accordingly. The RDF inference component is connected to the OWL-DL reasoner, which is responsible for classification and answering OWL-DL queries. The Racer system [27] is used as an OWL-DL reasoner.

Knowledge Management. Besides providing deductive support, DSs are responsible for collecting the status of SDOs, published in the OWL format, and building an integrated OWL description accessible to the reasoning process. The semantic representation of each SDO is added to the internal database of the DS. This semantic representation consists of a set of instances augmented with rules. Facts deduced from rules are only used to change parameters and to call services but never modify the knowledge base.

Architecture. OWL-SF is a distributed system an its functional architecture integrates two basic building blocks: OWL-SDOs and DSs. A system may be composed of multiple components of both types which can be added and removed dynamically at runtime. DSs use the SDO discovery and announcement implementation to get aware of new SDOs in the environment. Whenever a new SDO is discovered, its semantic representation is added to the internal database.

3.9 DRAGO

Distributed Reasoning Architecture for a Galaxy of Ontologies (DRAGO) [57] is a distributed reasoning system, implemented as a peer-to-peer architecture in

which every peer registers a set of ontologies and mappings, and the reasoning is implemented using local reasoning in the registered ontologies and by coordinating with other peers when local ontologies are semantically connected with the ontologies registered in other peers. DRAGO is implemented to operate over HTTP and access ontologies and mappings published on the web.

Types of Context. DRAGO does not implement a context layer, i.e. it does not have any service for context collection, storing or distribution.

Ontologies. DRAGO considers a web of ontologies distributed among a peer-to-peer network. Each peer may contain a set of different ontologies describing specific domains of interest (for example, ontologies describing different activities of users in a university). These ontologies may differ from a subjective perspective and level of granularity. In each peer there are also semantic mappings defining semantic relations between entities belonging to two different ontologies. These semantic mappings are described using C-OWL [9]. To register an ontology at a peer the users specify a logical identifier for it, i.e an URI, and inform a physical location of the ontology in the web. Besides that, it is possible to assign semantic mappings to the ontology, providing, in the same manner, the location of the mappings on the web. New peers may be added dynamically to the system, providing new ontologies and semantic mappings.

Inferences/Reasoning techniques. The reasoning process may compare concepts in different ontologies to check concept satisfiability, determining if a concept subsumes the other (i.e. the latter is less general than the former), based on the semantic mappings relating both ontologies. In a set of ontologies interconnected with semantic mappings, the inference of concept subsumption in one ontology (or between ontologies) may depend also on other ontologies related to the previous ones through those mappings. Every peer registers a set of ontologies and mappings, and provides reasoning services for ontologies with registered mappings. Each peer may also request reasoning services from other peers when their local ontologies are semantically connected (through a mapping) with the ontologies registered at the other peer. The reasoning with multiple ontologies is performed by a combination of local reasoning operations, internally executed in each peer for each distinct ontology. A distributed tableau algorithm is adopted for checking concept satisfiability in a set of interconnected ontologies by combining local (standard) tableaux procedures that check satisfiability inside the single ontology. Due to the limitations of the distributed tableau algorithm, for a semantic mapping DRAGO supports three types of rules connecting atomic concepts in two different ontologies: *is equivalent*, *is subsumed* and *subsumes*. A Distributed Reasoner was implemented as an extension to the open source OWL reasoner Pellet [59].

Knowledge Management. As each peer registers sets of heterogeneous ontologies and mappings, the knowledge base is totally distributed. When users or applications want to perform reasoning with a registered ontology they refer to the

corresponding peer and invoke its reasoning services giving the URI to which the ontology was bound.

Architecture. DRAGO aggregates a web of ontologies distributed amongst a peer-to-peer network in which each participant is called a *DRAGO Reasoning Peer* (DRP). A DRP is the basic element of the system and is responsible for providing reasoning services for ontologies using the semantic mappings registered. As these mappings establish a correlation between the local ontology and ontologies assigned to other DRPs, a DRP may also request reasoning services of other DRPs as part of a distributed reasoning task. A DRP has two interfaces that can be invoked by users or applications. A *Registration Service Interface* is available for creating/modifying/deleting registrations of ontologies and mappings assigned to them. A *Reasoning Service Interface* enables requests of reasoning services for registered ontologies. Among the reasoning services DRAGO allows to check for ontology consistency, build classifications, verify concepts satisfiability and check entailment.

3.10 Conclusion

In this section, we classify the surveyed systems according to our taxonomy (cf. Table 2), and discuss their suitability for implementing context-oriented ontological reasoning for Ambient Intelligence. The eight systems we presented not only have different features, but some of them have been developed with different purpose. Gaia, CoBrA, Semantic Spaces and CHIL offer middleware infrastructure for Smart Spaces; SAMOA is designed specifically to support applications that deal with social networks in ubiquitous environments; CAMUS provides an infrastructure for the development and execution of a network-based intelligent robot system; OWL-SF supports the design of generic distributed context-aware systems; finally, DRAGO provides reasoning about heterogeneous ontologies.

Comparing the four frameworks for Smart Spaces, it may be said that Gaia is the only one that supports distributed knowledge bases and is the one that best deals with dynamic scenarios, allowing context providers to be added or removed dynamically and ontologies to be dynamically modified with regard to types of context and their properties. Despite not being tailored specifically for smart spaces, OWL-SF may be used for implementing such systems, as its singular characteristic is its support for distributed inference. Similar to Gaia, OWL-SF considers a distributed knowledge base. Hence, in each space, the aggregated context information will depend on the available providers, avoiding communication bottlenecks and allowing more efficient information processing and dissemination. The disadvantage of this approach is that context consumers cannot know beforehand which context information will be available at each space, and that it may happen that the necessary information may not be available.

While most systems have no mechanism to deal with heterogeneity of context representation through different spaces, CAMUS and DRAGO pay attention to this subject. CAMUS has its ontology structured in layers to provide a shared vocabulary as an approach to tackle the problem, while DRAGO is the only that

	Set of context providers	Ontology update	Knowledge base	Main goal of reasoning	Handling of heterogeneous knowledge bases
Gaia	Dynamic	Dynamic	Distributed	Derive higher-level facts	No
CoBrA	Dynamic	Static	Centralized	Both	No
Semantic Spaces	Dynamic	Static	Centralized	Derive higher-level facts	No
CHIL	Static	Static	Centralized	Derive higher-level facts	No
SAMOA	Static	Dynamic	Distributed	Derive higher-level facts	No
CAMUS	Dynamic	Dynamic	Centralized	Derive higher-level facts	Shared vocabulary layer
OWL-SF	Dynamic	Static	Distributed	Both	No
DRAGO	—	Dynamic	Distributed	Classification	Semantic Mapping

Table 2. Classification of middleware systems for context-oriented ontological reasoning.

supports the inclusion of generic mappings between the ontologies. However, DRAGO does not provide a context infrastructure, i.e. it does not have any service for context acquisition and distribution. In fact, it is solely dedicated to support reasoning with heterogeneous ontologies.

AmI applications are composed of independent entities that act autonomously in an open-ended environment, driven by their own goals. In order to fulfill their tasks, collaboration with peers is often required. Different entities are very likely to employ different knowledge representations; therefore the ability to align such representations into a single one that can be shared by different applications is paramount to ensure communication. DRAGO architecture, presented in this section, relies on pre-defined mappings to align different ontologies. Nevertheless, in practical implementations of AmI it is not feasible to build in advance mappings of all possible pairs of different ontologies that may be needed. There

are other techniques to overcome the barrier of heterogeneous representations in such conditions. The next section is dedicated to a survey of approaches that try to solve exactly this problem.

4 Approaches for Ontology Alignment

Entities acting autonomously in an open-ended environment will often require collaboration with peers to fulfill their goals. Because different entities are likely to provide separate ontologies, the ability to integrate the ontologies into a single representation is paramount to ensure overall communication. This need, often referred to the *ontology alignment* problem [35] consists in finding a set of equivalence between a set of nodes in ontology A and a set of nodes in ontology B (see Figure 1). More formally, the problem of ontology alignment can be compared to that of database schema matching. Given two schemas, A and B, one wants to find a mapping m from the concepts in A into the concepts of B in such a way that, for all $(a, b) \in A \times B$, if $a = \mu(b)$, then b and a have the same meaning. Several approaches have been proposed to perform such alignments. They can be organized into three categories [35]: structural methods (which rely only on the structure of the ontology and the nodes labels), instance-based methods (which compare the instances of each concept in the ontologies) and methods based on a reference ontology which acts as a mediator. This field is wide and complex[3], but its application to the interaction of entities in ubiquitous environments leads to the specification of a sub-category of problems:

- The alignment process must be performed *on the fly* and in a *limited amount of time*. Indeed, in open systems, it is not possible to know in advance the nature of the entities that interact, which makes impossible to compute in advance the alignment of their ontologies.
- The entities that interact share common goals or common capacities. Thus, one can consider in most applications that the intersection of ontologies will not be empty. As a consequence, there always exists an acceptable alignment between two ontologies. However, one cannot take for sure that concepts will appear at the same level of specialization. For instance, one ontology can have a single class for the concept of *research paper*, while the other directly works with the sub-concepts *journal*, *conference_proceedings*, etc.
- The ontology alignment must be performed automatically (whereas a lot of work in this domain relies on semi-automatic approaches). As a consequence, entities must decide on alignments without the validation of a human expert. Thus, they must be able to evaluate the trust they have in the resulting alignment, e.g. by valuating the equivalence links depending on their ambiguity.

The next subsection present the lexical alignment (a.k.a *anchoring*) that is used as a basis by all ontology alignment approaches. We then present the

[3] See http://www.ontologymatching.org for a complete description.

three main approaches for ontology alignment (structural, instance-based and mediation-based). We illustrate the advantages and drawbacks of each technique and a brief overview of most significant work in each category. Subsection 4.5 then presents a brief overview of semantic similarity measures and how they can offer a new solution for ontology alignment.

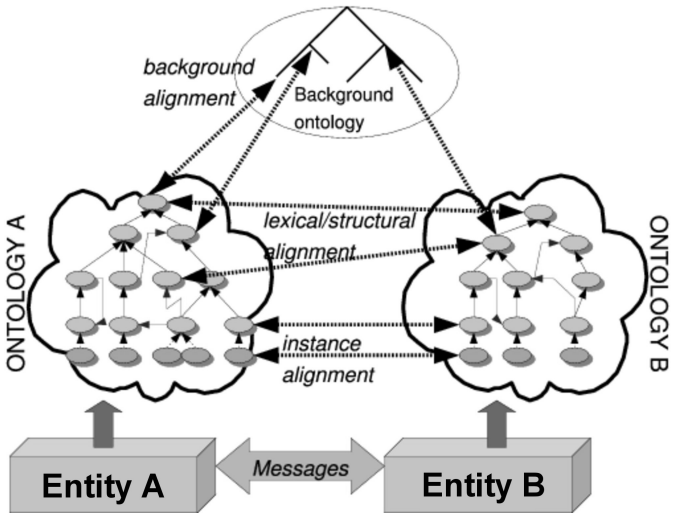

Fig. 1. Ontology alignment is a set of equivalence between nodes of both ontologies. This schema presents the three classical solutions: alignment based on the structural properties, alignment based on the concepts instances and alignment based on a "background ontology".

4.1 Lexical Alignment

Lexical anchoring is, generally, the first processing step of ontology alignment tools. It is possible to differentiate several kinds of approaches, with advantages and drawbacks. First, classical Natural Language Processing tools, as lemmatization (which constructs singular or infinitive forms of words, for instance, determining that *kits* is the plural of *kit*, *bought* is a derived form of *buy*), tokenization (which considers each word of a compound concept, like *long_brain_tumor subClassOf long_tumor* [3]) or suffixprefix approach (which searches in a sub-part of the words. For instance, like *net* is an abbreviation of *network*, *ID* can stand for *PID*). However, these approaches have some limitations: the lemmatization can be ambiguous (out of the sentence context, *left* can be lemmatized either into *left:adjective* or *leave:verb*); the tokenization requires to choose the correct sub-concepts inference (is *brain_tumor subClassOf brain* a valid association?); and the prefixsuffix alignment is strongly dependant of the language (for instance,

hotel should not match *hot*, nor can *word* be seen as an abbreviation of *sword*). For these reasons, the lexical anchoring has to be used with great care and to be completed and/or confirmed with other techniques.

A complementary approach of all these methods is the lexical distance measure, so called "edit distance" between two strings (Hamming distance or Levenhstein distance). For example, the Levenhstein distance is given by the minimum number of operations needed to transform one string into the other, where an operation may be an insertion, deletion, or substitution of a single character. It is widely used for spell checking. The main advantages of edit distance are that it reproduces NLP approaches when words are not too much complex. For instance, the translation from plural to the singular form has a cost of 1 in most words (removing the trailing "s"). However, some drawbacks still remain, like *sword* is equivalent to *word*, which has a cost of 1 and could be wrongly accepted.

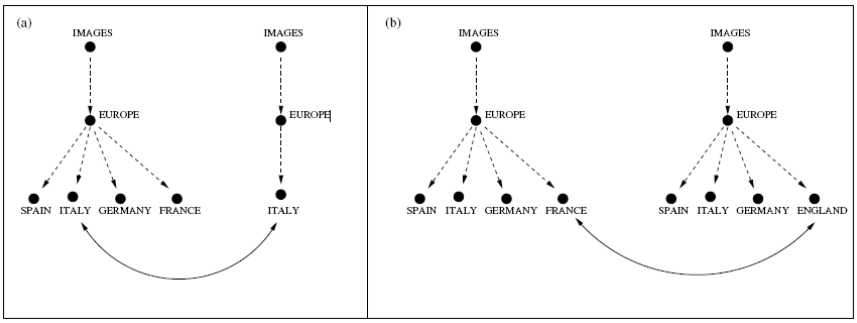

Fig. 2. Structural alignment error example based on hierarchy analysis in Ctx-Match [10].

4.2 Structural Approaches

Structural approaches are based on the structural comparison of the two concepts graphs (in the meaning of graph theory). It relies on lexical anchoring as a first step for associating lexically-close labels from both ontologies. The complementary alignment pairs are obtained by an extended hierarchy comparison around these anchored concepts (e.g. in CATO [14], the authors make use of a specific algorithm for three comparison (so-called TreeDiff) to find the largest common substructure between trees. The CATO system will be presented in more details in Subsection 5.4). More generally, such structural methods will match terms like PC and Personal Computers when sub-classes and properties described the same concept (like ID, model, etc.). However, structural alignment may fail if the information is not classified using the same criterion or if the ontologies do not cover the same fields or instances. As Figure 2a shows, the

concept “Italy” from the ontology to the right will be correctly aligned with the concept “Italy” from the ontology to the left, because they share lexically-close concepts in their whole hierarchical structures. But in Figure 2b, the concept “England” from one ontology will be wrongly aligned with the concept “France” from the other ontology, because, although they do not have the same meaning, they also share lexically-close concepts in their whole hierarchical structures.

4.3 Instances-based approaches

The objective of these methods is to determine an alignment using common instances between the two ontologies. When the common instances are identified, the main idea is to suppose that the hierarchy declares theses instances under the same concepts (maybe structurally or lexically different). For example, in [30], the authors tried to align the categories hierarchy of Google and Yahoo. An instance is identified using the URL of websites. Regarding to [63], the positive and/or negative matches of instances between two concepts allows them to compute subsumption alignment, in addition of equivalence alignment. For example, in Figure 3, if instances of *roadvehicle* concept of ontology 2 are classified as instances of *vehicle* concept of ontology 1, and the opposite is not true, then it is possible to deduce that *vehicle* is a super-class of *roadvehicle*. If instances of *van* concept of ontology 1 are classified as instances of *roadvehicle* concept of ontology 2, and not the opposite, *roadvehicle* is identified as a super-class of *van*.

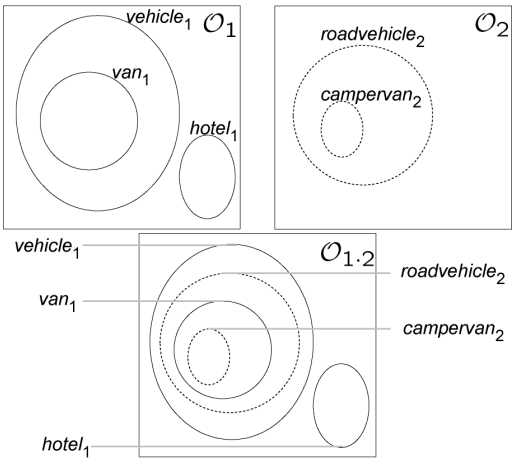

Fig. 3. Instance-based alignment [63], permits to construct subsumption alignment in addition of equivalence alignment.

However, the main drawback of this approach is the instances detection. For example, the work by Ichise et al. comparing Yahoo and Google hierarchy only generates 10% of common instances. Moreover, in van Diggelen work, it

is difficult to conclude if instances intersection is not complete (i.e. if one class does not contain all instances of another class), even if it is just a problem of miss-definition of concepts in one of the two ontologies.

4.4 Mediated approaches

Mediated approaches are based on the use of a third ontology to mediate the alignment process (see for instance [4, 10][4]). The main advantage of these methods is to be more robust in case of ontologies that differ greatly either lexically or structurally, or when no instances are provided. For example, in [3], the authors align two ontologies with very different formalisms, which could not be done using a structural approach. Thus, one of the major prerequisite (which is also the major limit of the approach) is to have access to a mediator-ontology with enough of information to anchor concepts from the two initial ontologies on it. The anchoring stage is generally a lexical anchoring as presented in the previous section. After the anchoring stage, the two ontologies are represented by two set of concepts from the mediator ontology. For example, in Figure 4, following the "is more general" relation allows the authors to find an alignment between "Aorta thoracalis dissection" (ontology A) and "Dissection of artery" (ontology B). The main difficulty in mediated approaches is to define a strategy to construct semantic paths between these two sets, using the structure of the mediator ontology.

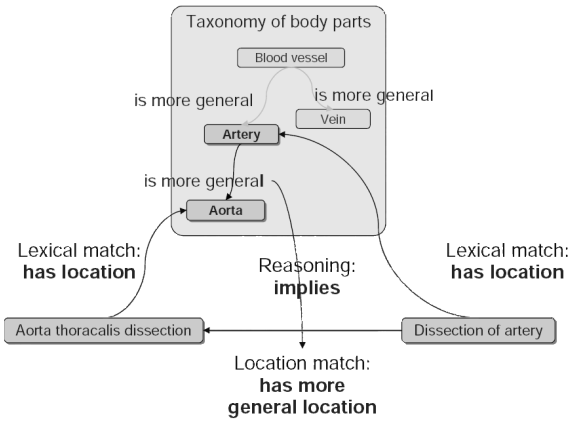

Fig. 4. Mediated alignment approach. Example from [3].

[4] Work from [10] is essentially structural. However, they also use WordNet [22] as mediator to solve some ambiguity.

4.5 Alignment based on semantic similarity

Finding paths between concepts in an ontology is at the core of mediated approaches but it can also be used to complete lexical and structural alignments. For instance, a given ontology concept may not be directly attached to an application-defined concept, as required for context interpretation. This case may happen, for instance, if the alignment was difficult, or if the ontology is large. Thus, we can use *semantic similarity measure* on the entity ontology (as it is done on the background ontology in mediated approaches) to compute correct semantic paths and to valuate the strength of this path.

In this section, we first recall the general principle of semantic similarity and we then propose to use it within a mediated approach for aligning two entity ontologies.

Semantic similarity. From a theoretical point of view, semantic similarity is a formula that allows users to evaluate the amount of common attributes shared by two concepts. Different kinds of approaches were proposed, based on a concepts hierarchy (e.g. [15]), on glosses from a dictionary (e.g. [6, 43, 48]), etc. In this paper, we will focus in similarity for ontology, since it is suitable with our initial problem of interaction between entities.

Work on semantic similarity with ontologies can be splitted in two major methods: the *edge-based* approach and the *node-based* approach. The edge-based approach [49] makes use of the shortest taxonomic path to define the semantic similarity between two concepts. The main idea behind is intuitive: in a semantic taxonomy, the longer the path, the less semantic similar are the two concepts. Recent work in this area has focused on the issue of weighting the edges, which permits to refine the value of a semantic link (e.g. [29, 31, 70, 71]).

The node-based approach [51] is the most used nowadays and is considered to be the most efficient for the semantic similarity ([15] provides a good survey on the subject). The weight of a node represents the information content of the concept. In other words, the more general a concept is (i.e. near from the root), the less information it contains. There exist different kinds of formulae that combine the information content of the two target nodes and their closest common parent. The *closest common parent* is the node that is the most specific in the set of common ancestors nodes (e.g. [32, 37, 38]).

Semantic-similarity based alignment. In Aleksovski work (Section 4.4), the use of mediator ontologies is limited to a reduced set of patterns of paths, which are considered to be "semantically correct". Thus, two concepts can only be related with a binary relation (the alignment exist or it does not exist). In the framework of interacting entities, we suggest that it is necessary to have a solution to valuate the strength of an alignment. This weight will be useful to solve ambiguity and to propose more complex dialogue strategies, as proposed in [41].

The mediator ontology should be either WordNet [22] (especially for human/agent communication) or the ontology of a third mediator agent if this

ontology contains some common concepts of the two other ontologies. A first lexical anchoring of terms within the mediator ontology is performed, using the edit distance of Levenshtein. Then, the system computes the set of all possible semantic score between each concept from the set of anchored concept of the first ontology to the set of anchored concept of the second ontology.

The key problem in this approach is that a real ontology inherently contains a lot of different relation types. To tackle this problem, we have proposed a measure of semantic relatedness [42], which is more general than similarity measure [51], to take into consideration the entire graph and not only the hierarchy. The preliminary evaluation of our measure, applied to human-machine communication, emphasized that our correlation factor with human judgment is approximately 20% better that other measures.

4.6 Conclusion

Ontology alignment is a key issue in open systems that require some form of distributed reasoning, such as in open Ambient Intelligence. While most middleware systems currently use a single ontology, openness and heterogeneity require distributed entities to be able to interpret information derived from other peers, based on an a priori unknown ontology. The three main approaches for ontology alignment we presented (structural, instance-based and mediation-based) all contain some limitation. While the structural methods are the most efficient ones, they require that the ontologies be very similar (e.g. two ontologies derived from a single initial specification). The instance-based approaches offer the consistency of Description Logic inference rules, but they work only if each concept is associated with a complete set of instances (e.g. document URIs). Mediation allows aligning very heterogeneous ontologies (even the knowledge representation formalisms can differ), but the background ontology is generally very large (i.e. larger than each one of the mediated ontologies).

5 The Campus Approach

In this section, we discuss Campus, a framework for the development of Ambient Intelligence applications [56]. Based on multi-agent systems technology, Campus provides an infrastructure to develop innovative context aware applications that accommodate mobile devices and environment sensor devices. The Campus architecture is intended as a configurable framework in which users can decide what services they want to enable in their environments, rather then a monolithic application. It is composed of three levels: the context-provisioning layer, the communication and coordination layer, and the ambient services' layer, as illustrated by Figure 5. In a nutshell, the bottom level is responsible for offering basic middleware services and functionalities, such as providing context information and device discovery. The communication and coordination layer offers support for semantic interoperability, providing discovery, exchange and collaboration among hybrid entities, regardless of proprietary representations of

information. Finally, the topmost layer provides application specific and ambient services and acts as a hotspot, i.e., allows users to extend the framework by plugging in specific services required by a particular user, environment, type of collaboration, of interest to their environment.

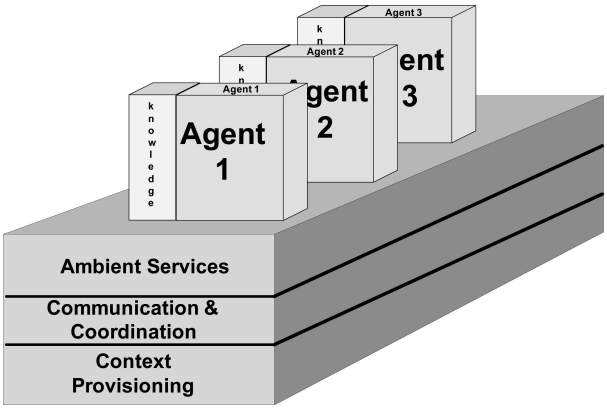

Fig. 5. Abstract view of Campus architecture.

As shown in Figure 6, agents distributed through the two bottom layers implement the main functionalities of Campus. In the context-provisioning layer, *Context Monitor Agents* (CMA) collect raw context data from various sources such as devices, sensors and applications, and make it available for interested entities. *Distributed Reasoning Agents* (DRA) infer and disseminate higher-level context information. In each smart space a *Local Knowledge Agent* provides persistent knowledge storage. It aggregates the context information obtained from context providers (i.e. CMAs and DRAs) available in that area, and builds a partial ontological view. The LKA may be queried by entities interested in finding context providers in that area. In the communication and coordination layer, a *Knowledge Interoperability Agent* (KIA), is responsible for semantic alignment of ontologies. It will provide this information to LKA whenever needed. In the further part of this section, the main features of Campus are discussed separately.

5.1 Context Types

In the Campus framework, context data comprises not only information about mobile devices, users' preferences and roles, description of institutional physical spaces, but also data collected from personal and smart spaces applications (e.g. appointments in a personal agenda, list of activities in an organizational scheduler, etc). *Context Monitor Agents* (CMA) are responsible for collecting raw context data from various sources such as sensors, devices and user applications; interpreting it as context information according to a predefined ontology; and making this information available for interested entities.

Most information about mobile devices is acquired with the aid of the MoCA middleware [53], a component that supports the development of context-aware applications and services for mobile computing. MoCA provides efficient services to collect context information associated with mobile devices (e.g. CPU usage, available memory, battery level, etc). This information comprises not only raw data related to the device's resources and the wireless links (currently, only IEEE 802.11), but also the symbolic location of each device, which is inferred from the RSSI values measured at the device with respect to all the Wi-Fi Access Points in the devices vicinity. On the other hand, much context data is obtained from applications and files that store personal and organizational information. For example, a CMA running on the notebook of professor Silva could make available information about Silva's agenda and preferences, and also about the Silva's notebook. A CMA agent running on a fixed computer at LIP6 could collect data about the schedule of activities, worker profiles, etc.

5.2 Ontologies

The Campus upper ontology serves as a knowledge base for the framework implementation, i.e., provides the necessary semantics to allow high-level exchanges, including brokering, negotiation and coordination amongst software entities. It contains precise definitions for every relevant concept in the framework, e.g., it defines that context providers and services are described by a tuple containing its name, a parameter list, a capability list, the communication port number, and protocol. Of course, the concepts of name, parameter list, capabilities list, port and protocol are also defined in the ontology. This ontology serves as a static model of our domain and will be used as a basis upon which mediation services will try to reason and understand the information provided by entities in the environment.

5.3 Reasoning

In Campus, we propose a distributed reasoning mechanism to infer and disseminate higher-level context information, i.e. context information that may be deduced using data obtained from other context providers. In our approach, each reasoning element is called a *Distributed Reasoning Agent* (DRA). A DRA is able to deduce new knowledge reasoning about description logic rules. In such rules, atoms that depend on some context data compose the antecedent of the rule, while the consequent of the rule defines a new piece of context information. CMAs collect context information from several sensors or obtain it from applications and database files that contain user's preferences, device's descriptions or specific data, such as the list of activities scheduled for a set of rooms, and made it available for DRAs. Facts are deduced in runtime, described according with the respective ontology, and updated in the knowledge base. DRA implements also an event-based communication interface to where other entities subscribe their interest about a high-level context (defined by a rule). These entities are notified whenever the state context satisfies a given rule [64].

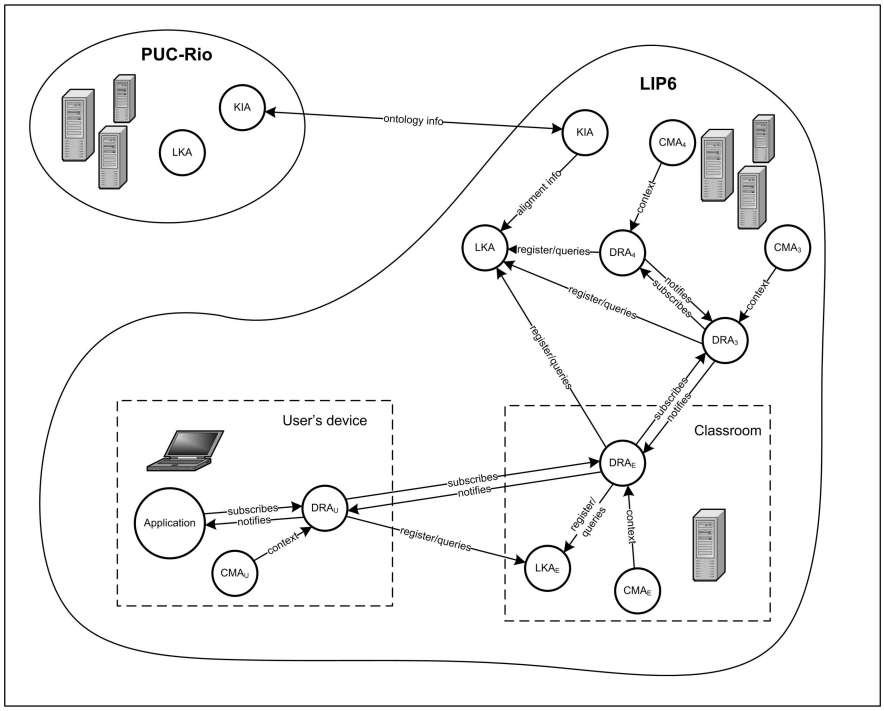

Fig. 6. Campus multi-agent architecture.

Since we consider a fully distributed environment, some context information necessary for processing the logical inference of a given rule may not be directly available from CMAs for a DRA. In such case, the DRA will subscribe at other DRAs, which are capable of providing (by inference) the required piece of context information. In this sense, each DRA acts simultaneously as a provider, a reasoner and a consumer of context knowledge. This distributed approach brings several advantages. It allows the distribution of the high computational cost of the inferences process. As each DRA may have a different "context view", depending on the device on which it is running and its location, not all items need to be kept at a single database, avoiding a communication bottleneck. Besides, distributed inference may hide private data still revealing context information that may be inferred from it.

Figure 6 shows an example of this distributed interaction in the scenario where Mr. Silva enters a classroom to attend a meeting with the Campus team. In this case, Silva's smart phone executes DRA_U (user's DRA) that has access to the data obtained from the sensors at the smart phone (sound, luminosity, movement), its own location data and some administrative data available for Silva. As he enters the room, i.e. changes its location, DRA_U initiates a discovery process, and as a result, it detects the presence of LKA_E and then another reasoner, DRA_E, which is responsible for accessing the room's sensors, storing

their context data and doing ambient-specific context reasoning. We assume that an application at the smart phone responsible for managing the ring tone has already subscribed at DRA_U to get control notifications for the ring tone adjustment according the following rule:

$$\textbf{Device(?d)} \wedge \textbf{isLocatedIn(?d,?e)} \wedge \textbf{ClassroomInUse(?e)} \Rightarrow \textbf{InSilenceMode(?d)}$$

While a CMA that wraps a location service delivers the smartphone's binary property "isLocatedIn" to DRA_U, the "ClassroomInUse" unary property is only made available by the DRA_E responsible for the classroom. This property is inferred from the following rule:

$$\textbf{Environment(?e)} \wedge \textbf{hasScheduledActivity (?e, ?a)} \wedge \textbf{ClassActivity(?a)} \wedge \textbf{ActivityOncourse (?a)} \wedge \textbf{LecturerPresent(?e)} \Rightarrow \textbf{ClassroomInUse(?e)}$$

Moreover, supposing that DRA_E has no direct access to the activities' time schedule and time reference, nor to the location data of every person in the institution, it has to rely on context knowledge inferred by another two agents: DRA_3 to monitor the "ActivityOnCourse" unary property and DRA_4 to obtain notifications of the "LecturerPresent" unary property. Then, DRA_3, running on an "activity manager" (i.e. fixed device running a CMA dedicated to monitor the time and the schedule of activities) will process the following rule:

$$\textbf{Activity(?a)} \wedge \textbf{startTime(?a, ?t1)} \wedge \textbf{finishTime(?a, ?t2)} \wedge \textbf{presentTime(?t3)} \wedge \textbf{isLessThan(?t1, ?t3)} \wedge \textbf{isBiggerThan (?t2, ?t3)} \Rightarrow \textbf{ActivityOncourse (?a)}$$

At the same time, DRA_4, running on a "location manager" which has access to location information from all mobile devices detected in the building, will process the following rule:

$$\textbf{Device(?d)} \wedge \textbf{isLocatedIn(?d,?e)} \wedge \textbf{isCarriedBy(?d,?p)} \wedge \textbf{playsRole(?p, ?r)} \wedge \textbf{Lecturer(?r)} \Rightarrow \textbf{LecturerPresent(?e)}$$

In fact the reasoning outcome of DRA_U will be the result of the cascading reasoning, with new context data being inferred initially by DRA_4.

It is straightforward to notice that for interactions among DRAs referencing different ontologies, it is necessary to provide an intermediating agent that has the ability to resolve – or at least, try to resolve – the semantic mismatch among nodes in the different ontologies. For example, when the DRA_U, which is a foreign entity, wants to obtain the "ClassroomInUse" context data, and this same information is represented as "RoomBusy" in DRA_E, then the intermediate agent will have to support the interaction between the entities identifying the identity of "ClassroomInUse" and "RoomBusy" in the scope of this particular application.

5.4 Ontology Alignment

In this section, we will consider the problem of ontology alignment (as defined in Section 4) in the Campus architecture. To enable the automation of the process, we coded the resource representations using W3C's OWL-DL ontology standard, for the reasons previously discussed in Subsection 2.3. Ontologies are expressive, formal, machine processable representations that fulfill the knowledge requirements of AmI applications.

Our past experience with semantic interoperability enabled us to provide CATO [11, 13, 14, 21], a solution that combines well known algorithmic solutions, e.g., natural language processing, the use of similarity measurements, and tree comparison, to the ontology alignment problem. We propose to incorporate CATO to the kernel of the Campus framework. The philosophy underlying CATO's strategy mixes syntactical and semantic analysis of ontological components. Its current implementation combines the lexical and structural approaches discussed in Subsections 4.1 and 4.2 respectively.

During the alignment, lexical and structural comparisons are performed in order to determine if concepts in different ontologies should be considered semantically compatible. A refinement approach is used that alternates between lexical and structural comparison between ontological concepts. The process begins when concepts from both ontologies go through a lexical normalization process, in which they are transformed to a canonical format that eliminates the use of plurals and gender flexions. The concepts are then compared, with the aid of a dictionary. The goal is to identify pairs if lexically equivalent concepts.

We assume that lexically equivalent concepts imply the same semantics, if the ontologies in question are in the same domain of discourse. For pairs of ontologies in different domains, lexical equivalence does not guarantee that concepts share the same meaning [45, 61]. To solve this problem, we adopted a structural comparison strategy. Concepts that were once identified as lexically equivalent are now structurally investigated. Making use of the intrinsic structure of ontologies, a hierarchy of concepts connected by subsumption relationships, we now isolate and compare concept sub-trees. Investigation on the ancestors (superconcepts) and descendants (sub-concepts) will provide the necessary additional information needed to verify whether the pair of lexically equivalent concepts can actually be assumed to be semantically compatible.

Lexical comparison is done during the first and second steps of the strategy. Structural analysis is done in the second and third steps of the strategy. The final result is a OWL document containing *equivalent class* statements (<owl:equivalentClass>) that relate the equivalent concepts from the two input ontologies. This is equivalent to a mapping between conceptual schemas. The proposed strategy is depicted in Figure 7.

First Step: Lexical Comparison. The goal of this step is to identify lexically equivalent concepts between two different representations, as presented in Subsection 4.1. We begin by assuming that lexically equivalent concepts are

also semantically equivalent in the domain of discourse under consideration, an assumption that is not always warranted.

Each concept label in the first ontology is compared to every concept label present in the second one, using lexical similarity as the criteria. Filters are used to normalize the labels to a canonical format: (i) If the concept is a noun, the canonical format is the singular masculine declination; (ii) if the concept they represent is a verb, the canonical format is its infinitive. Besides using the label itself, synonyms are also used. The use of synonyms enriches the comparison process because it provides more refined information. For example, in the scenario proposed in Subsection 1.1 of this chapter, the "activity", "class" and "meeting" concepts were identified as synonyms in our database.

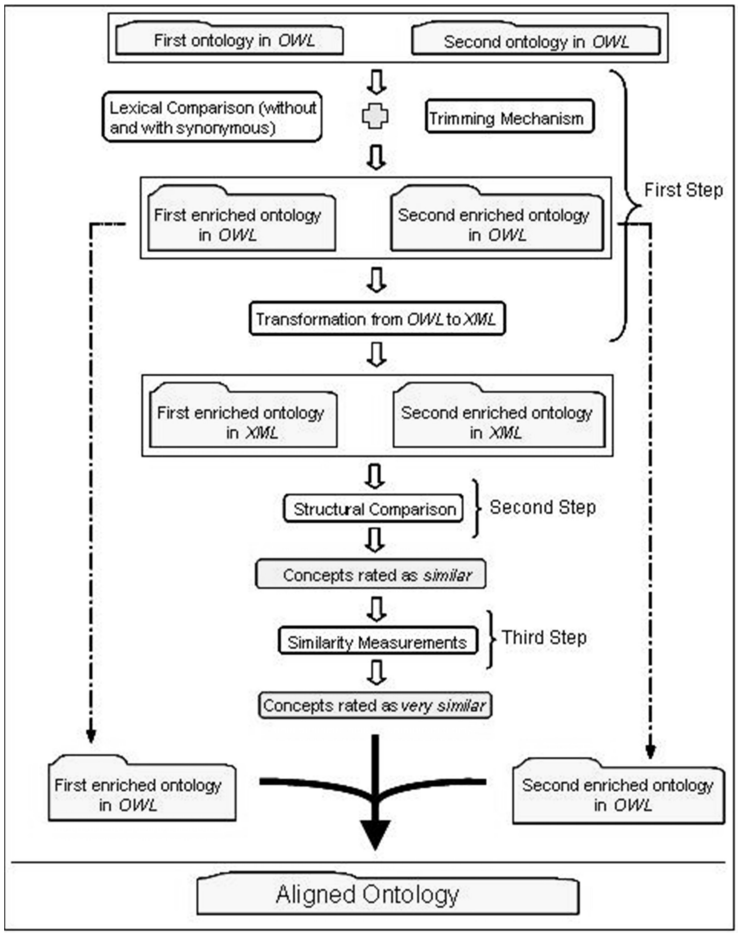

Fig. 7. CATO ontology alignment strategy [13].

Lexical similarity alone is not enough to assume that concepts are semantically compatible. We also investigate whether their ancestors share lexical similarity. It is important to note that the alignment strategy in this step is restricted to concepts and properties of the ontology. As a result of the first stage of the proposed strategy, the original ontologies are enriched with synonyms and links that relate concepts that are known to be lexically equivalent.

Second Step: Structural Comparison Using *TreeDiff*. Comparison at this stage is based on the subsumption relationship that holds among ontology concepts, similarly to what was discussed in Subsection 4.1, not taking into consideration ontology properties and restrictions. Our approach is thus more restricted than the one proposed by Noy and Musen [45], that analyses the ontologies as graphs, taking into consideration both taxonomic and non taxonomic relationships among concepts.

Because we only consider lexical and structural relationships in our analysis, we are able to make use of well-known tree comparison algorithms. We are currently using the *TreeDiff* [65]. Our choice was based on its ability to identify structural similarities between trees in reasonable time.

The goal of the *TreeDiff* algorithm is to identify the largest common substructure between trees, described using the DOM (*Document Object Model*) model [5]. This algorithm was first proposed to help detect the steps, including renaming, removing and addition of tree nodes, necessary to migrate from one tree to another (both trees are the inputs to the algorithm).

The result of the Tree Diff algorithm is the detection of concept equivalence groups. They are represented as subtrees of the enriched ontologies. Concepts that belong to such groups are compared in order to identify if lexically equivalent pairs can also be identified among the ancestors and descendants of the original pair. Differently from the first step, where we based our analysis and compared concepts that were directly related to one another, we are now considering the structural vicinity of concepts. Every concept in the equivalence group is investigated in order to determine lexically equivalent pairs, number of matching sons, number of synonymous concepts in the sub-trees, available from the previous step, and ancestor equivalence.

Third Step: Fine Adjustments based on Similarity Measurements. The third and last step is based on semantic similarity measurements, as discussed in Section 4.5. In CATO, concepts are rated as very similar or little similar based on pre-defined similarity thresholds. We only align concepts that were both classified as lexically equivalent in the second step, and thus rated very similar. Thus the similarity measurement is the deciding factor responsible for fine-tuning our strategy. We adapted the semantic similarity measurement strategies proposed in [40]. The similarity threshold is fixed by the users, and can be adjusted to enforce a firmer similarity policy.

During this step the "Activity" and "Class" concepts, from the visiting professor scenario, are aligned with the "Talk" and "Lecture" concepts belonging

to the Campus upper ontology. Those concepts were rated equivalent during the second step. Their similarity level is calculated in the third step.

The final ontology, containing mappings between concepts imported from the two input ontologies, will provide a common understanding of the semantics represented by both input ontologies. This representation can now be shared by entities searching for information, seeking to discover or to compose with other AmI applications. Code 1 depicts a part of the output ontology for this example.

In Campus, the ontology alignment is implemented by the *Knowledge Interoperability Agent* (KIA), which is responsible for applying the CATO strategy for determining the equivalent classes between different ontologies, whenever foreign entities come to interact together. If an entity queries the context infrastructure looking for some context information represented using a different ontology, it will resort to KIA to obtain the equivalent class in the prevalent ontology.

Code 1 :*Ontology alignment*

```
<owl:Class rdf:ID="Activity">
  <rdfs:subClassOf rdf:resource="#CATO_Thing"/>
  <owl:disjointWith rdf:resource="#Event"/>
  <owl:equivalentClass>
    <owl:Class rdf:about = "Talk">
  </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="Lecture">
  <rdfs:subClassOf rdf:resource="Educational_Activity"/>
  <owl:equivalentClass>
    <owl:Class rdf:about = "Class">
  </owl:equivalentClass>
</owl:Class>
```

The main contribution of CATO's strategy is to combine well-known algorithmic solutions, such as natural language processing and tree comparison, to the ontology integration problem. CATO is fully implemented in Java and relies on the use of the JENA API. The use of the API helped us focus on the alignment process, for it made ontology manipulation transparent. JENA reads and filters information from the tags of files written in an ontology language and transforms it to an abstract data model in which ontological concepts can be manipulated as objects.

6 Conclusion and Open Problems

Design and operation of open Ambient Intelligence Environments pose several huge challenges to the research community, which are caused mainly by the inherently heterogeneous, distributed and dynamic nature of these systems. In this chapter we first surveyed, analyzed and classified several middleware systems that propose partial solutions to the corresponding complex problem of distributed context reasoning. It turned out that only some of the systems support distributed context reasoning, and in fact only two tackle the problem

of managing heterogeneous context knowledge. Then, we discussed the main general approaches for semantic alignment of ontologies, as this is a basic requirement for coping with heterogeneous knowledge representations. Finally, we presented our approach for distributed reasoning and semantic alignment in the scope our ongoing effort to develop a multi-agent based framework Campus for development of Ambient Intelligence.

Within the Campus framework we focused on the provision of distributed context reasoning – using Distributed Reasoner Agents (DRAs) – and the construction of a software component responsible for the automatic alignment of ontologies – the Knowledge Interoperability Agent (KIA). Our strategy is based on the application of well-known software engineering strategies, such as lexical analysis, tree comparison and the use of similarity measurements, to the problem of ontology alignment. Motivated by the requirements of AmI applications, we proposed an ontology alignment strategy and tool that produces an ontological representation that makes it possible for such applications share common understanding over information available on environment [69].

6.1 Discussion and Future Work

Building complex Ambient Intelligence environments requires the integration of several different context providers, which may be dedicated sensors, user's applications, databases monitors, etc. The inclusion of new types of context providers will require the implementation of new Context Monitor Agents with specific interfaces and functionalities.

The distributed inference of high-level context information brings the advantage of sharing the complexity of the reasoning process among several devices, allowing quicker response times. But on the other hand it requires efficient context dissemination to work efficiently. Aiming for efficient performances, the balanced distribution of DRAs among the devices that compose Ambient Intelligence typical scenarios is an issue to be investigated.

Any automated ontology alignment solution presents some degree of risk, in the sense that it cannot fully guarantee that the most adequate equivalence between concepts will be always identified. Limitations of the algorithms used, time to perform the computations, and possible lack of information coded in the original ontologies may, in some of the cases, prevent the automated solution to identify answers that would be otherwise manually found. The success of the CATO approach depends on the volume and quality of the information coded in the input ontologies. The richer and more complete the information, the better the results. Conversely, if the input ontologies are poorly defined, incomplete or lacking, the ontology integration engine has little data to work upon, and thus is not likely to deliver adequate results.

To tackle such situations, an alternative solution may be the use of an instance-based approach, as presented in Subsection 4.3. Each implementation of a device, such as Dr. Silva's smart phone (SMP-1) or notebook (NTB-1) can be thus represented by an ontology, containing a set of classes, restrictions, properties (data schema), that corresponds to the internal knowledge representation

of each device. The goal of the instance-based approach is the same, i.e., find matching classes across different ontologies.

The instance based approach uses a query probing technique that consists of exhaustively sending keyword queries to original ontologies [66]. Further analysis of the results using learning algorithms and statistical analysis provides indication of good matches. This approach can be generalized to any domain that provides a reliable substitute for an unique instance identifier. In the Geographic Information systems domain, for example, there are various geo-referencing schemes that associate geographic object with a description of its location on the Earth's surface. This location acts as a universal identifier for the object, or at least an approximation thereof. We have successfully applied this approach to build mediators for Geographic Data Catalogs [11, 12, 24]. We are currently adapting the approach to be part of the Campus Framework kernel and help improve CATO results.

The CATO semantic adjustment makes use of the Maedche architecture [40], to confirm that a "very similar" rated alignment is semantically correct (and not only lexically and structurally). However, as stated by Maedche himself, the semantic measure used has two limits: 1) it only uses the taxonomic information from the ontology; 2) it does not consider that two different given edges in a taxonomy do not carry the same information content (as demonstrated in [51], see Subsection 4.5). We have proposed in [42] a new measure of semantic relatedness, which considers different weight for edges and different edges types. The preliminary evaluation of our measure shows that it increases approximately by 20% the correlation factor with human judgment. We currently try to integrate this measure in a refinement of the Maedche algorithm, in order to enhance the semantic adjustment in the CAMPUS framework.

References

1. G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, and P. Steggles. Towards a Better Understanding of Context and Context-Awareness. In *Proc.of the 1st international symposium on Handheld and Ubiquitous Computing (HUC 99)*, pages 304–307, London, UK, 1999. Springer-Verlag.
2. J. Ahola. Ambient Intelligence. Final report, IST Advisory Group, February 2001.
3. Z. Aleksovski, M. Klein, W. ten Kate, and F. van Harmelen. Matching Unstructured Vocabularies using a Background Ontology. In *Proc. of Knowledge Engineering and Knowledge Management (EKAW)*, pages 182–197, 2006.
4. Z. Aleksovski, W. ten Kate, and F. van Harmelen. Exploiting the structure of background knowledge used in ontology matching. In P. Shvaiko, J. Euzenat, N. Noy, H. Stuckenschmidt, R. Benjamins, and M. Uschold, editors, *Proc. of First International Workshop on Ontology Matching (OM-2006), co-located with the 5th International Semantic Web Conference (ISWC-2006), Athens, Georgia, USA*, CEUR Proceedings, November 2006.
5. V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A.L. Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood. Document Object Model (DOM) Level 1 Specification. Recommendation, 1998.

6. S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 805–810, 2003.
7. A. Bikakis, T. Patkos, G. Antoniou, and D. Plexousakis. A Survey of Semantics-based Approaches for Context Reasoning in Ambient Intelligence. In R. Bergmann, K.-D. Althoff, U. Furbach, and K. Schmid, editors, *Proceedings of the Workshop "Artificial Intelligence Methods for Ambient Intelligence" at the European Conference on Ambient Intelligence (AmI'07)*, pages 15–24, November 2007.
8. D. Bottazzi, R. Montanari, and A. Toninelli. Context-Aware Middleware for Anytime, Anywhere Social Networks. *IEEE Intelligent Systems*, 22(5):22–32, 2007.
9. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing Ontologies. In *Proc. of the Second International Semantic Web Conference (ISWC-2003)*, volume 2870 of *Lecture Notes in Computer Science*, pages 164–179. Springer, 2003.
10. P. Bouquet, L. Serafini, and S. Zanobini. Semantic Coordination: A New Approach and an Application. In Dieter Fensel, Katia P. Sycara, and John Mylopoulos, editors, *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2003.
11. D. Brauner, M.A. Casanova, and R. Milidiú. Mediation as Recommendation: An Approach to Design Mediators for Object Catalogs. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *OTM Workshops (1)*, volume 4277 of *Lecture Notes in Computer Science*, pages 46–47. Springer, 2006.
12. D. Brauner, C. Intrator, J.C. Freitas, and M.A. Casanova. An Instance-based Approach for Matching Export Schemas of Geographical Database Web Services. In L. Vinhas and Antnio C. R. Costa, editors, *Proc. of the IX Brazilian Symposium on GeoInformatics. Porto Alegre : Sociedade Brasileira de Computao (GeoInfo)*, pages 109–120. INPE, 2007.
13. K. Breitman, D. Brauner, M.A. Casanova, R. Milidiú, and A.G. Perazolo. Instance-Based Ontology Mapping. In *Proc. of the Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems EASe 2007*, pages 117–126. IEEE Computer Society Press, 2007.
14. K. Breitman, C. Felicíssimo, and M.A. Casanova. CATO - A Lightweight Ontology Alignment Tool. In Orlando Belo, Johann Eder, Joo Falco e Cunha, and Oscar Pastor, editors, *CAiSE Short Paper Proceedings*, volume 161 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.
15. A. Budanitsky and G. Hirst. Evaluating WordNet-based Measures of Semantic Distance. *Computational Linguistics*, 32(1):13–47, 2006.
16. G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Department of Computer Science, Dartmouth College, 2000.
17. H. Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, University of Maryland, Baltimore County, December 2004.
18. H. Chen, T. Finin, and A. Joshi. An ontology for context-aware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 2003.
19. A. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.
20. K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J.-C. Burgelma. Scenarios for Ambient Intelligence in 2010. Final report, IST Advisory Group, February 2001.

21. C. Felicíssimo and K. Breitman. Taxonomic ontology alignment - an implementation. In *Proceedings of Workshop em Engenharia de Requisitos (WER 2004)*, pages 152–163, 2004.
22. C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
23. R.T. Fielding and R.N. Taylor. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, 2002.
24. A. Gazola, D. Brauner, and M.A. Casanova. A Mediator for Heterogeneous Gazetteers. In *Proc. of the XXII Brazilian Symposium on Databases (SBBD), Poster Session.*, volume 1, pages 11–14. Sociedade Brasileira de Computao, 2007.
25. A. Gómez-Pérez, M. Fernadéz-Peréz, and O. Corcho. *Ontological Engineering*. Springer Verlag, London, 2004.
26. D. Guan, W. Yuan, S.J. Cho, A. Gavrilov, Y.-K. Lee, and S. Lee. Devising a Context Selection-Based Reasoning Engine for Context-Aware Ubiquitous Computing Middleware. In J. Indulska, J. Ma, L. T. Yang, T. Ungerer, and J. Cao, editors, *UIC*, volume 4611 of *Lecture Notes in Computer Science*, pages 849–857. Springer, 2007.
27. V. Haarslev and R. Möller. RACER System Description. In *Proc. of the International Joint Conference on Automated Reasoning (IJCAR'01)*, volume 2083 of *Lecture Notes in Computer Science*, 2001.
28. K. Henricksen and J. Indulska. Modelling and using imperfect context information. In *Proc. Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*. IEEE Computer Society, 2004.
29. G. Hirst and D. St-Onge. *Lexical Chains as representation of context for the detection and correction malapropisms*, chapter 13, in WordNet: an electronic lexical database,. MIT Press, 1998.
30. R. Ichise, H. Takeda, and S. Honiden. Integrating multiple internet directories by instance-based learning. In G. Gottlob and T. Walsh, editors, *IJCAI*, pages 22–30. Morgan Kaufmann, 2003.
31. M. Jarmasz and S. Szpakowicz. Roget's thesaurus and semantic similarity. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *RANLP*, volume 260 of *Current Issues in Linguistic Theory (CILT)*, pages 111–120. John Benjamins, Amsterdam/Philadelphia, 2003.
32. J.J. Jiang and D.W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *Proc. on International Conference on Research in Computational Linguistics, Taiwan*, pages 19–30, 1997.
33. G. Jones. Challenges and opportunities of context-aware information access. In *UDM '05: Proceedings of the International Workshop on Ubiquitous Data Management*, pages 53–62, Washington, DC, USA, 2005. IEEE Computer Society.
34. M. A. Casanova K. Breitman and W. Truszkowski. *Semantic Web: Concepts, Technologies and Applications*. Springer Verlag, 2007.
35. Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: The State of the Art. In Y. Kalfoglou, M. Schorlemmer, A. Sheth, S. Staab, and M. Uschold, editors, *Semantic Interoperability and Integration*, number 04391 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2005.
36. H. Kim, Y. Cho, and S. Oh. CAMUS: A Middleware Supporting Context-aware Services for Networkbased Robots. In *Proc. of the IEEE Workshop on Advanced Robotics and Social Impacts*, 2005.
37. C. Leacock and M. Chodorow. *Combining local context and WordNet similarity for word sense identication*, chapter 11, in WordNet: an electronic lexical database, pages 265–283. MIT Press, 1998.

38. D. Lin. An Information-Theoretic Definition of Similarity. In Jude W. Shavlik, editor, *Proc. of the 15th International Conference on Machine Learning (ICML 1998), Madison, Wisconson, USA, July 24-27, 1998*, pages 296–304. Morgan Kaufmann, 1998.
39. J. Lindenberg, W. Pasman, K. Kranenborg, J.Stegeman, and M.A. Neerincx. Improving service matching and selection in ubiquitous computing environments: a user study. *Personal Ubiquitous Computing*, 11(1):59–68, 2006.
40. A. Maedche and S. Staab. Comparing Ontologies: Similarity Measures and a Comparison Study. Internal report, Institute AIFB, University of Karlsruhe,, 2001.
41. L. Mazuel and N. Sabouret. Generic Command Interpretation Algorithms for Conversational Agents. In *IAT*, pages 146–153. IEEE Computer Society, 2006.
42. L. Mazuel and N. Sabouret. Degré de relation sémantique dans une ontologie pour la commande en langue naturelle. In *Plate-forme AFIA, Ingnierie des Connaissances 2007 (IC 2007)*, pages 73–83, 2007.
43. S. Mohammad and G. Hirst. Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings, 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, Sydney, Australia, July 2006.
44. B. Mrohs, M. Luther, R. Vaidya, M. Wagner, S. Steglich, W. Kellerer, and S. Arbanowski. OWL-SF - A Distributed Semantic Service Framework. In *Proc. of Workshop on Context Awareness for Proactive Systems (CAPS), Helsinki, Finland*, pages 67–78, 2005.
45. N. Noy and M. Musen. The PROMPT suite: Interactive tools for ontology merging and mapping. *Int. J. Hum.-Comput. Stud.*, 59(6):983–1024, December 2003.
46. P. Nurmi and P. Floreen. Reasoning in Context-Aware Systems, 2004.
47. I. Pandis, J. Soldatos, A. Paar, J. Reuter, M Carras, and L. Polymenakos. An ontology-based framework for dynamic resource management in ubiquitous computing environments. In *Proceeding of the 2nd International Conference on Embedded Software and Systems (ICESS 2005)*, pages 1–8. Northwestern Polytechnical University of Xi'an, PR China, December 2005.
48. S. Patwardhan and T. Pedersen. Using WordNet-based context vectors to estimate the semantic relatedness of concepts. In *Proc. of the EACL 2006 workshop, making sense of sense: Bringing computational linguistics and psycholinguistics together. Trento, Italy*, pages 1–8, 2006.
49. R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 19, pages 17–30, 1989.
50. A. Ranganathan and R. Campbell. A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. *Lecture Notes in Computer Science*, 2672:143–161, January 2003.
51. P. Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proc. of IJCAI*, pages 448–453, 1995.
52. M. Román, C.K. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt. A Middleware Infrastructure for Active Spaces. *IEEE Pervasive Computing*, 1(4):74–83, October-December 2002.
53. V. Sacramento, M. Endler, H.K. Rubinsztejn, L.S. Lima, K. Gonalves, F.N. Nascimento, and G.A. Bueno. MoCA: A middleware for developing collaborative applications for mobile users. *IEEE Distributed Systems Online*, 5(10), 2004.
54. S. Sameshima, J. Suzuk, S. Steglich, and T. Suda. Platform Independent Model (PIM) and Platform Specific Model (PSM) for Super Distributed Objects. Final adopted specification OMG document number dtc/03-09-01, Object Management Group, September 2003.

55. M. Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.
56. A.F. Seghrouchni, K. Breitman, N. Sabouret, M. Endler, Y. Charif, and J.-P. Briot. Ambient intelligence applications: Introducing the campus framework. In *Proc. 13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2008), Dublin*. IEEE Computer Society Press, 2008.
57. L. Serafini and A. Tamilin. DRAGO: Distributed Reasoning Architecture for the Semantic Web. In Asuncin Gmez-Prez and Jrme Euzenat, editors, *ESWC*, volume 3532 of *Lecture Notes in Computer Science*, pages 361–376. Springer, 2005.
58. A. Shehzad, H.Q. Ngo, K.A. Pham, and S.Y. Lee. Formal modeling in context aware systems. In *Proceedings of the First International Workshop on Modeling and Retrieval of Context*, September 2004.
59. E. Sirin and B. Parsia. Pellet: An OWL DL Reasoner. In Volker Haarslev and Ralf Mller, editors, *Description Logics*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, June 2004.
60. J. Soldatos, N. Dimakis, K. Stamatis, and L.Polymenakos. A Breadboard Architecture for Pervasive Context-Aware Services in Smart Spaces: Middleware Components and Prototype Applications . *Personal and Ubiquitous Computing Journal*, 2007.
61. G. Stoilos, G. Stamou, and S. Kollias. A string metric for ontology alignment. In *Proceedings of International Semantic Web Conference (ISWC 2005)*, pages 624–637, 2005.
62. T. Strang and C.Linnhoff-Popien. A context modeling survey. In *First International Workshop on Advanced Context Modelling, Reasoning And Management*, Nottingham, England, September 2004.
63. J. van Diggelen, R. Beun, F. Dignum, R. van Eijk, and J. Meyer. Combining normal communication with ontology alignment. In *Proceedings of the International Workshop on Agent Communication (AC'05)*., volume 3859 of *LNCS*, 2005.
64. J. Viterbo, M. Endler, and J.-P. Briot. Ubiquitous service regulation based on dynamic rules. In *Proc. 13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2008), Dublin*. IEEE Computer Society Press, 2008.
65. J. Wang. An Algorithm for Finding the Largest Approximately Common Substructures of Two Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):889–895, 1998.
66. J. Wang, J.-R. Wen, F. H. Lochovsky, and W.-Y. Ma. Instance-based Schema Matching for Web Databases by Domain-specific Query Probing. In M. A. Nascimento, M. T. zsu, D. Kossmann, R.J. Miller, J. A. Blakeley, and K. B.Schiefer, editors, *VLDB*, pages 408–419. Morgan Kaufmann, 2004.
67. X.H. Wang, J.S. Dong, C.Y. Chin, S.R. Hettiarachchi, and D.Q. Zhang. Semantic Space: An Infrastructure for Smart Spaces. *Pervasive Computing*, 3(3):32–39, July-September 2004.
68. X.H. Wang, D.Q. Zhang, T. Gu, and H.K. Pung. Ontology based context modeling and reasoning using OWL. In *Proceedings of 2nd IEEE Conf. Pervasive Computing and Communications (PerCom 2004), Workshop on Context Modeling and Reasoning*, pages 18–22, Orlando, Florida, March 2004. IEEE Computer Society Press.
69. A. Williams, A. Padmanabhan, and M. Brian Blake. Local consensus ontologies for B2B-oriented service composition. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS '03)*, pages 647–654. ACM, 2003.

70. Z. Wu and M.S. Palmer. Verb Semantics and Lexical Selection. In *Proc. of the 32nd. Annual Meeting of the Association for Computational Linguistics (ACL 1994)*, pages 133–138, 1994.
71. J. Zhong, H. Zhu, J. Li, and Y. Yu. Conceptual Graph Matching for Semantic Search. In *Proceedings of the 10th International Conference on Conceptual Structures ICCS 2002*, pages 92–106, London, UK, 2002. Springer-Verlag.