

# Enhancing Semantic Interoperability in Collaborative Systems

Flavio De Paoli and Marco Loregian  
DISCo, University of Milano-Bicocca, Italy  
{depaoli, loregian}@disco.unimib.it

## Abstract

*A key successful factor for a knowledge management system is to promote and support semantic communication among people. This goal can be accomplished by integrating personal knowledge management tools with sharing tools that allow for merging semantic perspectives to create unified views, adapting behaviors to context, and finally learning from conversations to promote knowledge diffusion and acquisition. CADO is an open framework that provides for different levels of integration to support collaboration among members of an organization.<sup>1</sup>*

## 1. Introduction

Workers in a modern organization need to organize, share, and retrieve their personal knowledge. Many knowledge management (KM) systems suggest the use of ontologies to support the process of classification by means of metadata associated with documents. An issue is that such ontologies are often personal and need to be integrated to let groups of people share (part of) their knowledge. This knowledge exchange may occur on permanent bases (e.g., among people working on the same

project or belonging to the same community of interest) or occasionally (e.g., during formal meetings or informal conversations).

In the project MILK, a solution was developed to provide such kind of workers with a centralized environment to let them share ontologies, documents and documents' profiles [4, 6]. The goal of this paper is to move from a centralized system to a peer-to-peer one that lets a worker create his/her personal installation of the MILK KM system and then integrate it with the ones of the other workers. The CADO framework (*Context-aware Applications with Distributed Ontologies*, [7]) provides for the basic mechanisms and policies to address these issues. According to the MILK features, information navigation and retrieval across knowledge bases is accomplished according to the *context* in which it occurs. The *context* is a situation composed of a technological and social environment, and application objectives. This is derived from a well-known definition of context: “the location, identity and state of people, groups and computational and physical objects” [3].

We are interested in clusters of users that can communicate directly or remotely by means of their devices. In the former case, they share the same physical space and perceive the same environment. In the latter, they do not share a physical space, therefore they are in what we call a “virtual room”, which means that participants can concur in the creation of a common context even when being in different locations.

A cluster of workers in a particular room for a project meeting is an example of direct communi-

---

<sup>1</sup>This paper was published in: Proceedings of the Eighteenth International Conference on Software Engineering and Knowledge Engineering (SEKE 2006), pages 29—34. Published by Knowledge System Institute Graduate School, USA, 2006. This *draft* is provided for educational purposes only, all rights reserved.

cation that can occur by wireless connections (e.g., Bluetooth and WiFi). This communication is augmented by the CADO framework that lets them share the knowledge stored in their laptops. CADO enforces semantic interoperability to provide users with rich contextualized descriptions to promote awareness of available knowledge related to people and documents involved in the cluster.

The context is explicitly defined by a set of different (but interconnected) ontologies, each one describing a specific aspect that participates in the definition. For example, the context could be defined by involved devices, social goals, communication channels, people’s roles and interests. Note that we have said “could” instead of “is” to outline that CADO allows the user or the application to define what a context is by selecting the interesting topic ontologies among the available ones. The merging of such ontologies identifies the actual context model on which the application can rely.

The next section describes the CADO architecture. Section 3 discusses the semantic interoperability issues. Section 4 is dedicated to related works and finally Section 5 draws some conclusion and outlines future works.

## 2. Framework architecture

The CADO framework defines a set of distributed components (*peers*) that can connect with each other, according to their type, to form *clusters*. Peers belonging to a cluster can run on the same *node* (i.e., the same computer) or be spread over different nodes. Components are classified according to the layered architecture described in Figure 1

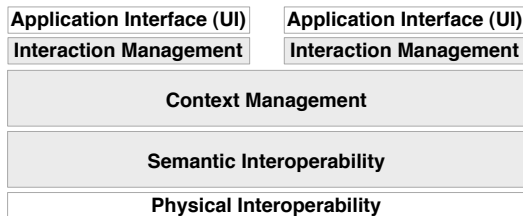


Figure 1. Framework architecture.

that assumes two computers running all the needed components (e.g., two laptops of collaborating colleagues). Every layer provides for interoperability at different level of abstraction, from physical to conceptual. The *physical* interoperability layer lets the two nodes communicate by addressing issues like message dispatching, fault tolerance, and user authentication. The *semantic* interoperability layer provides support for data processing by providing an integrated view over the involved ontologies. The *context* interoperability layer supports the selection of data and behavior according to specific situations. Finally, the *interaction management* layer supports the definition of the application logic and of the actual interfaces and interaction.

Primary components in CADO are *Interaction Managers (IM)*, *Context Managers (CM)*, *Ontology Managers (OM)* and *Ontology Adapters (OA)*. The set is completed by additional components that supply specific services such as the MILK Metadata Management System (MMS) components [6], and Lightweight-Directory-Access-Protocol (LDAP) components.

Figure 2 shows, with a sample situation, how peers can connect and how they are layered. The interaction layer is composed of interaction managers that support various devices (some with high computing capabilities, like PCs and laptops, and some with low capabilities, like digital pens and mobile phones). Interaction managers are connected to context managers. Context managers are connected to ontology managers that in turn are connected to ontology adapters. At different layers, peers are connected to form a multi-tier application (vertical connections). At every layer, peers can

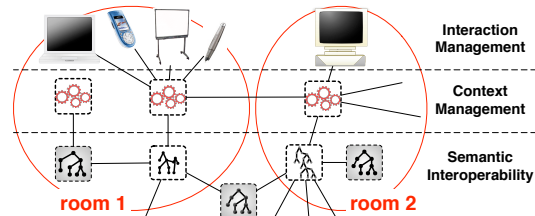


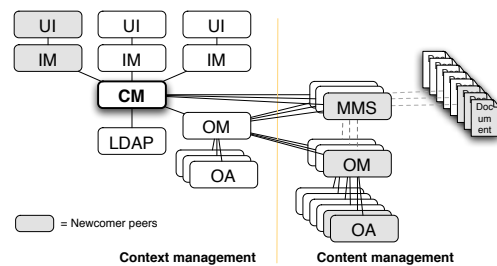
Figure 2. Sample configuration.

communicate and coordinate to each other (horizontal connections). Oval lines indicate clusters of peers forming (virtual) rooms. In room 1, we assume four devices (a laptop, a mobile phone, a large screen, and a digital pen) that share the same context and are coordinated by the same context manager. This could be the situation in a meeting room where workers, carrying some personal devices, are discussing in front of a large screen (e.g., a Smart whiteboard). Room 2 is an example of remote peers.

**Interaction Managers** are user agents with application-specific or device-specific capabilities that provide users with interaction features and perform cluster discovery and set up. An IM controls the user interface layout and the content information displayed according to context information supplied by context managers. Inside a (virtual) room, IMs behave according to the technological and social perspective. For example, an IM controls the large screen that acts as a shared display in the face-to-face meeting in room 1 (Figure 2).

**Context Managers** enact clustering mechanisms and define specific “policies” to support the behavior of the participants. Such policies are specified by a set of ontologies [7]. In the example of Figure 2, the four devices in room 1 are coordinated by the central CM according to its perception of the environment. CMs are contacted by IMs when they fall in range according to casual discovery protocol (e.g., the one provided by JXTA technology) or when two or more persons decide to cluster. IMs and CMs communicate to each other to establish the *context* for the cluster, and elect a coordinating CM that will be in charge of routing information among participating IMs. Connection and cluster capabilities are constrained by device capabilities. Devices with advanced connectivity support and high performance computing power (e.g., PCs, laptops) take care of simpler devices (e.g., sensors, RFID readers, digital pens, interactive screens) by driving their behavior to supply users with rich interactive environments.

A clustering example is shown in Figure 3, in which a newcomer actor joins a cluster. The joining protocol can be summarized as follows [7]: when



**Figure 3. Example of cluster.**

an IM gets in contact with a set of CMs, peers configurations are exchanged so that the IM knows the available technological contexts (i.e., where CMs are located and the IMs they already cluster). The IM selects a CM according to application policies (or under explicit direction of the user) and the user is authenticated by the CM contacting an LDAP peer (if required). The IM sends its configuration to the CM, specifying the MMS component in charge of managing user’s documents. The CM will provide an integrated view over all the involved MMSs supplying IMs with document profiles. Profiles include URIs through which IMs can retrieve the actual files. The MMS supplies the CM with information on its reference ontologies. Such ontologies will be integrated with the CM ontologies (Figure 3) according to the semantic interoperability features described in the next section.

### 3. Semantic interoperability

According to a well known definition, an *ontology* is an explicit specification of a conceptualization [9]. An ontology is composed of a hierarchical taxonomy, additional relations between concepts, concept descriptions (attributes), axioms (constraints) and instances. CADO users adopt ontologies to cover different aspects of the domains of interest, that can be independent (e.g., work practices and product specifications). We call **topic ontology** the part of an ontology that covers a particular aspect in a domain, collecting a set of concepts connected with a *is-a* relation and with other topic-specific relations (later referred to as *intra-topic re-*

lations). Therefore, a domain is covered by ontologies on different topics, that are interconnected via domain-specific relations (later referred to as *inter-topic relations*). In CADO, we consider *distributed ontologies* that are built from individual representations: “[a distributed] ontology is divided into several component ontologies and [...] each ontology [is constructed] individually (perhaps in parallel) by different developers in a distributed environment” [14]. In other words, distributed ontologies are modular and managed by peer components spread over a network.

Since knowledge is manifold, different users exploit different topic ontologies to cover their entire knowledge and expertise. Therefore, a collective vision over a topic within a group of individuals, such as a work team or a community, can only be achieved by merging (different) topic ontologies. Ontology merging is the “creation of [a] single coherent ontology that includes the information from all the sources” [12]. In CADO only *virtual* merging is employed, in that no physical ontology is produced by merging topic ontologies, but only a temporary *view* over different sources seen as a whole.

Given two (or more) ontologies to be merged, corresponding concepts in the taxonomies of the ontologies have to be identified: this requires a similarity measure for taxonomies (i.e., a matching criterion). When similarities are identified, the corresponding concepts have to be merged preserving relations — taxonomy *is-a*, concept-specific attributes, ... — and tracking origins to remember the source ontologies. In summary, according to literature [11], ontology merging consists of (1) finding places in the ontologies where they semantically overlap — either extensionally, i.e., concepts with the same label or synonyms, or intensionally, i.e., concepts with the same place in the taxonomy — (2) check the consistency, coherency and non-redundancy of the result. Iteration of such steps might be required.

In the current implementation [7], a basic ontology merging has been implemented as default mechanism. It starts with the merging of topic ontologies. For every topic ontology the activities are: (1) visit the taxonomy to identify overlapping

concepts; (2) built the new taxonomy including all the concepts in the original taxonomies; (3) add any other intra-topic relation to deliver the merged topic ontology. The ontology merging is then completed by adding inter-topic relations according to the original ontologies. During merging, simple checks are performed to avoid cycles and replications. Note that tackling the merging problems by considering topic ontologies ensure a higher degree of confidence, than considering generic taxonomies due to the high coupling of involved concepts. CADO allows for merging customization by overriding similarity and check procedures with tailored matching criteria (e.g., translations, synonym analysis, WordNet-mediated mapping).

In the CADO framework, the merging tasks are performed by ontology *managers* that rely on ontology *adapters* to access single modules. **Ontology Adapters (OA)** manage a single ontology module, such as an individual topic ontology; supply a standard interface to access the managed ontology as an OWL ontology (can be wrappers of existing ontologies); manage local ontology evolution. **Ontology Managers (OM)** accomplish the merging of different ontologies; rely on modules managed by OAs. Such components can be connected to form a hierarchy with OAs as terminal components. Ontology Managers identify and select candidate ontologies according to following policies:

- **Manual selection policy:** OMs are explicitly supplied with ontology URIs at runtime.
- **Static selection policy:** OMs are configured to merge a given set of ontologies, which are identified by URIs.
- **Automatic selection policy:** OMs encompass application logics to identify ontologies of interest.

Static selection policy could be adopted to refer to normative ontologies (e.g., standards, reference ontologies of an organization) and to personal ontologies. An automatic selection policy could be adopted for looking for specific topic ontologies (e.g., device and spatial ontologies) according to

the definition of context given by the context managers. The logics can be programmed by extending the default mechanisms or by supplying a sample taxonomy (e.g. the one adopted by the actual user of the system). In the latter case, the default is to look for some similar concepts (by exact matching) in candidate ontologies accessed by reachable OAs.

The described merging activity can affect also the merged ontologies that can *learn* from the others some new concepts. This mechanism allows for knowledge diffusion and promote ontology evolution. The process is managed by the OAs according to the following policies:

- **Normative policy:** an ontology can only be modified by authorized (usually human) administrators. This policy is usually adopted for reference ontologies (e.g., an organization's ontology) that are maintained off-line.
- **Selective policy:** an ontology can be modified on external requests (from OMs), but only by authorized users and possibly with explicit authorization from the owner. This policy could be adopted to deal with sensitive ontologies that reflect user's knowledge representation.
- **Plastic policy:** an ontology can be freely modified by OMs. This policy is adopted by flexible nodes that need to reflect external events. For example an ontology related to the social interactions should adopt such a policy.

#### 4. Related Work

A requisite for sharing is to make people able to understand each other, which means that every participant is asked to contribute to the creation of an accessible common knowledge base. The problem of achieving semantic interoperability among different actors (and their devices) by exploiting online resources (i.e., ontologies [5]) is a hot topic in literature [8], grounding for example in Artificial Intelligence research on ontology mapping and merging. Several tools and systems have been proposed for concept mappings and ontology merging. Popular examples are the PROMPT suite [12]

and Chimaera [11] that are semi-automatic merging tools. Important issues are consistency and resolution of semantic conflicts. Among possible strategies, we can mention the SCROL ontology [13] and collaborative design [10]. To tackle these problems, CADO provides only basic mechanisms (see Section 3) that can be replaced or customized to meet expected levels of accuracy and automatization.

From an architectural perspective, a good example of semantic interoperability is provided by Aberer et al. [1, 2] that proposes a framework addressing semantic agreement between peers as a local issue rather than global and centralized. Agreement is achieved by making expert users explicitly specify mappings between individual schema. The transitive propagation of manually expressed mappings is defined as *semantic gossiping*. Automatic processes to establish semantic mappings are not envisioned, whereas in CADO this option has been considered (e.g., according to [13]). Further, (portions of) topic ontologies are propagated gossip-style, in a selective way that does not necessarily require domain expert intervention. Moreover, the CADO users (and system designers) can specify policies to regulate the access to ontologies (for Ontology Adapters) and decide how ontologies have to be clustered (for Ontology Managers).

For space reasons, this paper does not cover the context management and application interoperability (i.e., interaction management) layers with details. Both of them have been designed to meet openness requirements. Openness at context management level means that how reasoning is enacted (e.g., by means of which inference engine, or constraint validation technique) is not strictly imposed by the framework, but it is also customizable (also according to policies adopted at ontology management level).

#### 5. Conclusion and Future Work

The CADO framework overcomes the traditional peer-to-peer frameworks that allow for file exchange and repository replication. CADO add semantic layers with the main purpose of enabling

knowledge circulation and, at the same time, to support personalized knowledge representation and organization. System designers can customize the default mechanisms of CADO to accommodate specific policies for clustering, context definition, ontology merging, profiling and retrieval of knowledge. A first level of customization can be achieved by means of configuration files that can be edited by the final users, a deeper customization can be achieved by means of plug-in techniques that enable for the replacement of the framework logics. The great advantage is the high flexibility of CADO that can fulfil a variety of requirements in a single and coherent environment.

The MILK system is an example of a new generation of applications with richer knowledge representation and advanced interaction protocols. The CADO framework can be considered to be an evolution of the MILK approach to encompass distributed representation of knowledge and more various and contextualized interaction patterns and information presentation. By decoupling information presentation from interaction, context, and knowledge base management, the framework is open to the design of new interactions and to the development of device-specific, as well as situation-specific interfaces.

Future work will finalize the current prototype that is based on JXTA p2p technology and Protégé/OWL ontology managers. The goal is to deliver a framework that can be used by researchers in different fields to experiment advanced solutions on the more important aspects: knowledge management (e.g., ontology management, knowledge navigation, knowledge retrieval), social interaction (e.g., collaborative work, knowledge sharing, rich visualization), and finally ubiquitous and pervasive computing (e.g., new devices, embedded devices, seamless transition between contexts).

**Acknowledgement** This work was partially supported by the MILK project (IST-2001-33165).

## References

- [1] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. A framework for semantic gossiping. *SIGMOD Rec.*, 31(4):48–53, 2002.
- [2] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. The chatty web: emergent semantics through gossiping. In *WWW '03*, pages 197–206, New York, NY, USA, 2003. ACM Press.
- [3] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *HUC '99*, pages 304–307, London, UK, 1999. Springer-Verlag.
- [4] A. Agostini, S. Albolino, R. Boselli, G. De Michelis, F. De Paoli, and R. Dondi. Stimulating knowledge discovery and sharing. In *GROUP'03*, pages 248–257. ACM Press, 2003.
- [5] H. Alani. Ontology Construction from Online Ontologies. In *WWW2006*, (in print) 2006.
- [6] R. Boselli, R. Dondi, and F. De Paoli. Knowledge organization and retrieval in the MILK system. In *SEKE 2003*, pages 372–376, 2003.
- [7] F. De Paoli and M. Loregian. Context-aware Applications with Distributed Ontologies. In T. Latour and M. Petit, editors, *Advanced Information Systems Engineering, 18th International Conference, CAiSE 2006, Luxembourg, June 5-9, 2006. Proceedings of Workshops and Doctoral Consortiums*, pages 869–883. Presses Universitaires de Namur, 2006.
- [8] M. Ehrig, S. Staab, and Y. Sure. Bootstrapping ontology alignment methods with APFEL. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *ISWC 2005*, volume 3729 of *LNCIS*, pages 186–200. Springer, 2005.
- [9] T. R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, 1993.
- [10] C. W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Commun. ACM*, 45(2):42–47, 2002.
- [11] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *KR2000*, pages 483–493, 2000.
- [12] N. F. Noy and M. A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *AAAI-2000*, pages 450–455. AAAI Press / The MIT Press, 2000.

- [13] S. Ram and J. Park. Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflicts. *IEEE Trans. Knowl. Data Eng.*, 16(2):189–202, 2004.
- [14] E. Sunagawa, K. Kozaki, Y. Kitamura, and R. Mizoguchi. An environment for distributed ontology development based on dependency management. In *ISWC 2003*, volume 2870 of *LNCS*, pages 453–468. Springer, 2003.