Journal of Universal Computer Science, vol. 14, no. 1 (2008), 27-58
submitted: 6/11/06, accepted: 17/10/07, appeared: 1/1/08 © J.UCS

Ontoolcole: Supporting Educators in the Semantic Search of CSCL Tools

Guillermo Vega-Gorgojo, Miguel L. Bote-Lorenzo
Eduardo Gómez-Sánchez, Juan I. Asensio-Pérez
Yannis A. Dimitriadis
(School of Telecommunications Engineering, University of Valladolid
Camino del Cementerio s/n, 47011 Valladolid, Spain
{guiveg, migbot, edugom, juaase, yannis}@tel.uva.es)

Iván M. Jorrín-Abellán
(Faculty of Education, University of Valladolid
Camino del Cementerio s/n, 47011 Valladolid, Spain
ivanjo@doe.uva.es)

Abstract: Collaborative learning systems can be constructed following the service-oriented computing paradigm. This allows educators to integrate external tools, offered as services by software providers, in order to support the realization of collaborative learning situations. Discovering appropriate services is a challenging task that requires the description of their capabilities. This can be accomplished with Ontoolcole, an ontology of collaborative learning tools designed with the aim of supporting educators in the search of CSCL tools. Ontoolcole is depicted here and some new, relevant features are discussed. Namely, Ontoolcole incorporates an artifact module, a task-level coordination module and the description of static information resources, further improving the capabilities to describe complex CSCL tools. As a proof of concept, we also present a preliminary prototype of the intended target application of Ontoolcole, an interactive system for the search of CSCL tools, named Ontoolsearch. A case study with practitioners has been carried out to evaluate whether Ontoolcole can be employed by educators to search CSCL tools. Evaluation results show that Ontoolcole abstractions fit educators' questions based on their real practice while retrieving useful tools for their educational needs.

Key Words: ontologies, semantic web services, service discovery, collaborative learning systems, CSCL tools
Category: H.3.3, H.4.3, I.2.4, K.3.1

1 Introduction

Computer Supported Collaborative Learning (CSCL) [Koschmann, 1996] is a mature research field of increasing interest in recent years that promotes the use of Information and Communication Technologies within the context of collaborative learning. The so-called *collaborative learning systems* (CLSs) typically offer an environment with several tools that users may employ to accomplish specific learning activities. These tools can be collaborative in order to support the realization of group tasks, such as an asynchronous discussion carried out with a bulletin board. However, not all of the activities involved in a CSCL situation need to be collaborative: some may require individual tools such as a text editor for the writing of a report.

CLSs must be very flexible in order to accommodate a wide range of learning situations given the diversity of curricula, teaching styles and cultural differences among institutions, or even educators at the same institution [Roschelle et al., 1999]. In this sense, most of the existing CLSs include generic tools such as chats or document repositories that can be useful in a wide range of learning situations [Betbeder, 2003]. However, more specific tools are needed to achieve particular pedagogical objectives, e.g. concept map tools to ease the understanding of new concepts by relating them to students' previous knowledge [Jonassen, 2006, ch. 10]. Other specific tools offer a precise functionality for a particular domain, such as C-CHENE [Baker and Lund, 1996], a collaborative system for learning energy concepts in Physics.

In summary, it is difficult that a CLS can anticipate all possible learning situations that educators may require. However, it is highly desirable that they can adjust a collaborative learning system to their specific needs. Such a feature is called *tailorability*, and allows users to modify a system functionality so as to better fit their needs [Morch, 1995]. In this sense, tailorable CLSs enable educators to easily integrate external tools in order to support the realization of new situations. *Service-oriented computing* [Papazoglou and Georgakopoulos, 2003] can help the development of tailorable CLSs, since an educator can tailor a service-oriented CLS to his setting by selecting adequate services among those offered by service providers [Bote-Lorenzo et al., 2004]. Besides, service-oriented computing can promote *educational software reuse* since an existing service can be employed to support different learning activities.

There are ongoing proposals of service-oriented learning systems in the literature, such as [Bote-Lorenzo et al., 2007b], [Friesen and Mazloumi, 2004] or [Xu et al., 2003]. Moreover, the recent specification IMS Tools Interoperability Guidelines [IMS, 2006] defines a series of recommendations for the integration of third-party tools in a learning system based on the service-oriented computing paradigm. However, service-oriented computing introduces the challenge of *service discovery* since consumers need to find the most appropriate services for their current needs among available ones. Current approaches for service discovery are commonly based on well-known registries that are used by service providers to publish service metadata while consumers can query them to find suitable services. In the context of CSCL, educators are expected to perform the search of services in order to tailor a CLS to support their collaborative scenarios. These services offer ready-to-use tools that are accessible by learners and teachers through user interfaces. Throughout this paper we will use the term *CSCL service* to refer to any service-based tool that may be discovered by educators and integrated in a CLS to support a CSCL activity. Note that within this view a CSCL service is not necessarily designed for learning nor it is required to be collaborative.

Standard service registries, such as UDDI [OASIS, 2004] in the popular Web Services architecture [Curbera et al., 2002], use *keyword-based searches* to retrieve services whose descriptions contain the keywords included in a query using Information Retrieval techniques [Baeza-Yates and Ribeiro-Neto, 1999]. As reported in the literature [Lassila, 1998, Paolucci et al., 2002, Zhang et al., 2005], keyword-based searches are prone to obtain unexpected responses since a term may have more than one distinct meaning. Moreover, relevant results may not be retrieved because the terms employed to describe them do not match the query keywords, even though their meanings were similar [Fensel, 2004, pp. 91-95]. Given this limitation, ongoing research in *Semantic Web Service* (SWS) technology [Paolucci and Sycara, 2003] promises to automate Web Service discovery allowing *semantic searches*, as well as automatic invocation and composition. A semantic search intends to gather information compliant with the semantics of the query [Guha et al., 2003]. A search engine can achieve this by exploiting *ontologies* [Gruber, 1993] that are used to explicitly formalize knowledge, enabling rich descriptions and robust information retrieval. Preliminary results show that using ontologies to semantically annotate services allow much more expressive queries and enhanced precision in service retrieval [Paolucci et al., 2002, Klein and Bernstein, 2004].

However, these proposals do not support *educators' required queries*. A service discovery facility should allow educators to determine which CSCL services can support a collaborative scenario. To enable it, they should query for the offered functionality using high-level and comprehensible abstractions for educators, including supported collaboration features. Our previous work [Vega-Gorgojo et al., 2006] proposed a preliminary ontology of CSCL tools with the aim of allowing educators to perform semantic searches of CSCL services. In this ontology tools can be described specifying the set of tasks that can be performed by an actor (probably playing a role). In this paper, we present an evolution of this ontology, named *Ontoolcole*, with augmented capabilities to describe complex CSCL tools (although simple tools can also be described in an easy way) such as archival storage properties or decomposable group tasks. We also deepen on the role of this ontology for CSCL service discovery in a service-oriented CLS, although it could also be applied in the more general context of a recommender system of learning tools for CSCL settings. Indeed, the design of Ontoolcole is driven by the search system that would use this ontology. As a proof of concept, we present here *Ontoolsearch*, a preliminary prototype of such a system intended for educators' use. Finally, in order to evaluate if educators can benefit from using Ontoolcole for the semantic search of CSCL tools, we have carried out a case study in which some practitioners formulated questions for searching tools derived from their real practice.

The rest of this document is organized as follows: section 2 identifies the requirements for the discovery of CSCL services and discusses the alternatives to support it, motivating the use of Ontoolcole. Section 3 presents an overview of Ontoolcole and depicts the new features introduced to overcome some previous limitations. Next, section 4 introduces Ontoolsearch, an Ontoolcole-powered interactive system for the search of CSCL tools. Then, section 5 evaluates whether Ontoolcole can be employed by educators to search CSCL tools. Finally, the main conclusions of the study are shown as well as current research work.

2 Discovery of CSCL Services

This section deals with the issue of discovering CSCL services. First, service descriptions are discussed, emphasizing the attributes employed for service discovery. Then, some requirements are defined for the search of CSCL services. It follows an analysis of alternatives for the discovery of CSCL services.

2.1 Service descriptions

"Services are autonomous, computational entities that can be used in a platform-independent way. Services can be described, published, discovered, and dynamically assembled for developing massively distributed, interoperable, evolvable systems" [Papazoglou, 2007]. In this sense, it is possible to construct a new application based solely on sets of interacting services offering well-defined interfaces. The Service-Oriented Architecture (SOA) [Papazoglou, 2003] is a logical way of designing a software system to provide services to either end-user applications or other services distributed in a network through published and discoverable interfaces. In the SOA paradigm, service providers host services and *publish* their descriptions in a well-known service registry. Next, clients query one of these registries to *find* appropriate services. Using the obtained information, clients can *bind* to a service and *invoke* it. Due to the composition property of services, a service aggregator can arrange a composite service by aggregating a set of services. Composite services may provide a more complex functionality, although clients would perform the same interactions (find, bind and invoke) as in the former case.

Within this general picture, a tool that encloses a coarse-grained functionality can be exposed as a CSCL service. Software developers can design either a monolithic CSCL service or a composite one, composed by other lower-level services. For instance, the Collaborative Network Simulation Environment (CNSE) [Bote-Lorenzo et al., 2007a] is a service-oriented application intended for Computer Networks education that comprises several services such as a simulation

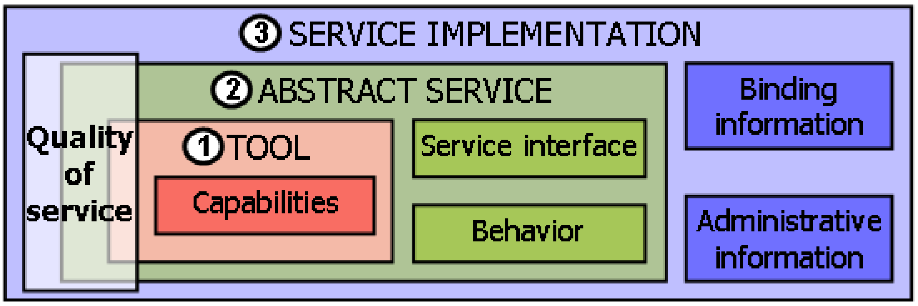

Figure 1: Level decomposition of a service. (1) The tool level defines the capabilities of a service, i.e. what a service does. (2) The abstract service level incorporates the interface and the behavior, defining how to interact with a service and its internal functioning. (3) The service implementation level defines a specific service endpoint in a hosting platform (the binding information) and administrative information of the provider. Finally, quality of service parameters define important functional and non-functional properties that involve the three levels

service or a visualization service. However, this aspect is irrelevant for a service client since the implementation is hidden to the users. Thus, once a CSCL service has been discovered and integrated in a service-oriented CLS, learners and educators can interact with it using a client that implements the presentation logic [Bote-Lorenzo et al., 2007b].

Concerning the discovery of CSCL services, the SOA *find* operation can be performed iteratively at different stages: initially, an educator searches for tool capabilities and obtains suitable CSCL services for his educational setting; the selection of a specific service binding is deferred, maybe based on other criteria such as quality of service parameters [Papazoglou, 2007]. To accomplish this iterative search, services need to be described with different kinds of metadata. Indeed, a service can be decomposed in three levels, each one described with different attributes, as shown in Fig. 1. The proposed model is based on the literature of services [Papazoglou and Georgakopoulos, 2003] and on the semantic approaches for the description of services OWL-S [Martin et al., 2004a], WSDL-S [Sheth et al., 2006] and WSMO [Roman et al., 2005].

At the tool level, the *capability* description states the conceptual purpose and expected results of the execution, i.e. what a service does. There is an `1:n` mapping between the tool and the abstract service levels, since different abstract services can be defined for the same tool. An abstract service is described by its *interface* and its *behavior*. The interface contains the operations and data types supported, and it is required to interact with a service. The behavior defines how the announced capabilities are achieved, i.e. its execution semantics. The behavior is commonly described with a workflow model and is mainly used for service composition. Lastly, a service implementation identifies a concrete service endpoint hosted in a provider's platform. There is also an `1:n` mapping between the abstract service and the service implementation layers. The *administrative information* describes the provider and the specific service, while the *binding information* specifies the access information required to actually invoke a service. Finally, the *quality of service* defines a series of quantitative functional and non-functional attributes that are assured by the service provider [Papazoglou, 2007] such as cost, security requirements, quality ratings or performance metrics, and are intended for service selection. Note that these quality of service properties affect all three levels since they involve implementation and deployment issues.

Thus, service discovery is mainly supported by the capability description, that should employ domain-specific concepts and taxonomies [Papazoglou and Georgakopoulos, 2003]. The following subsection discusses the requirements for the discovery of CSCL services, including the description of their capabilities.

2.2 Requirements for the discovery of CSCL Services

A service discovery facility should allow educators to find appropriate CSCL services for their learning scenarios. In order to be useful for educators, the search system should support their required queries. To determine the type of questions that educators command, we have carried out a preliminary study with eight CSCL practitioners. Specifically, we demanded them to pose questions for tools to support their real practice CSCL settings. Collecting *competency questions* is a common practice to gather user requirements [Staab et al., 2001], since they can help to determine user expectations of a system.

Table 2.2 shows an overview of some of the competency questions posed by educators. The analysis of these questions can serve to identify which properties should include the capability description of a CSCL service. Thus, questions CQ1 and CQ2 demand specific *tool types*, editor and document management tool. Remaining competency questions refer to required functionality of a tool using *task types* such as communication. Interestingly, all questions ask for *support of group work*, specifying collaborative tool or task types. The realization of group work is further refined in questions CQ3 and CQ5, defining synchronous and asynchronous *types of interaction*. Finally, some questions identify *artifact types*, such as concept maps or text documents, that are managed with a tool.

The capabilities of a CSCL service should describe the features identified in this study in order to properly support educators in the search of CSCL services. Noteworthy, other higher-level educational abstractions such as pedagogical objectives or type of knowledge to be learned could be employed to

Id	**Competency questions**
CQ1	I want a collaborative editor of concept maps
CQ2	I require a document management tool that can be used by groups of students to store and retrieve documents
CQ3	I want a synchronous tool for creating UML diagrams
CQ4	I want a tool to facilitate the communication among students
CQ5	I need a tool for editing text documents asynchronously

Table 1: Sample competency questions posed by educators for a search of CSCL tools

perform the search. Indeed, ongoing research on learning design tackles the definition of these aspects [IMS, 2003]. However, the description of a CSCL service should not be tied to a pedagogical scenario, since such a service can accommodate many different settings. In addition, a provider cannot anticipate all the possible scenarios in which technology can be used and describing them would add significant burden to providers. Thus, we have limited the description of CSCL service capabilities to the properties identified in this study that serve to define the offered functionality.

Besides, a service discovery facility can employ different technologies allowing different types of searches. Using traditional Information Retrieval techniques [Baeza-Yates and Ribeiro-Neto, 1999], both service descriptions and queries are modeled as sets of keywords. It allows *keyword-based searches* based on textual information, retrieving those services whose descriptions contain the keywords included in a user query. However, this type of search is prone to obtain irrelevant results while missing some relevant ones due to ambiguity of natural language [Fensel, 2004, pp. 91-95]. Broadly speaking, there are two related issues that we will call *synonymy* and *polysemy*. Synonymy reflects the fact that there are many ways to refer to the same concept, precluding the recall performance of a retrieval system [Deerwester et al., 1990]. In contrast, poor precision is affected by polysemy, since the same word can have different meanings.

To overcome these limitations of keyword-based searches, there are ongoing proposals to allow *semantic searches*. A semantic search denotes a concept about which the user is trying to gather information [Guha et al., 2003]. This way, a service discovery facility should understand the semantics of a query in order to properly respond. To achieve it, ontologies [Gruber, 1993] can be used to explicitly formalize a conceptual domain. An ontology could model the abstractions required by educators to search CSCL services and employed to semantically annotate them. A service discovery system could then use these annotations to unambiguously respond to semantic searches about CSCL services.

2.3 Approaches for the discovery of CSCL Services

Service discovery is commonly accomplished using well-known registries. For instance, UDDI [OASIS, 2004] is the service registry proposed for the popular Web Services architecture. Organizations can announce them in such a registry, as well as the services they offer. UDDI provides a white/yellow/green page functionality, allowing queries about organizations ("white pages"), services ("yellow pages") and technical information ("green pages"). However, UDDI is limited to keyword-based searches, precluding recall and precision, as discussed above. Moreover, UDDI and other existing service registries do not provide a capability representation language, as reported in [Martin et al., 2004b]. Thus, identified requirements for the discovery of CSCL services are not fulfilled with these registries.

SWS approaches like OWL-S [Martin et al., 2004a] and WSMO [Roman et al., 2005] make use of ontologies in order to support semantic searches of services. These proposals define mechanisms for the description of service capabilities, allowing capability matching of services. So far, OWL-S is perhaps the initiative that has attracted more interest. The OWL-S schema defines the element Service Profile to advertise a service, enabling its discovery. The profile describes the capabilities of a service in terms of the transformation produced. Specifically, it specifies the inputs required by the service and the outputs generated; as well as the preconditions required by the service and the expected effects that result from the execution of the service. Preliminary experiments with OWL-S provide implementations for service discovery based on matching the types of a service's inputs and outputs [Martin et al., 2004b, Kawamura et al., 2005]. However, the functionality of a service is not completely defined by the inputs and outputs, and other authors have proposed extensions to explain the relationship between inputs and outputs [Hull et al., 2006].

WSMO offers an alternative approach for service discovery, although with many similar characteristics to OWL-S [Lara et al., 2004]. Perhaps the main difference is the definition of user objectives as goals, which are decoupled from service capability descriptions. Both goals and capabilities are expressed by the state of world transition before and after service execution. A refinement of the WSMO approach for service discovery is explained in [Stollberg et al., 2007].

Concerning the discovery of CSCL services, OWL-S and WSMO would require extensions for the CSCL domain to describe them. Moreover, the majority of CSCL services involve cognitive processes that are difficult to express in terms of state transitions. Additionally, a CSCL service offers a coarse-grained functionality that should be described with other higher-level abstractions than input/output parameters.

In order to address aforementioned difficulties for the description of CSCL services, we have proposed the Ontoolcole ontology (see [Vega-Gorgojo et al., 2006] for a preliminary version). The depiction of this ontology is the topic of the following section. Ontoolcole aims to describe CSCL services at the tool level in order to support the discovery. Moreover, this ontology has been constructed taking into account educators' required questions identified in subsection 2.2.

3 Ontoolcole: an Ontology of Collaborative Learning Tools

Ontoolcole is an evolving ontology of collaborative learning tools designed with the aim of supporting educators in the search of CSCL tools. This section first provides an overview of Ontoolcole. Then, it depicts the new features introduced in Ontoolcole in order to overcome the three limitations found in a previous version [Vega-Gorgojo et al., 2006].

3.1 Overview of Ontoolcole

Technically, Ontoolcole is formalized in OWL DL [Bechhofer et al., 2004], a widespread and expressive ontology language with definite semantics that can be processed by a reasoner. A reasoner offers two valuable services for ontology development: ontology classification and consistency checking. When a reasoner classifies an ontology it generates the inferred ontology class hierarchy. The designer can then check whether the inferred class hierarchy reflects his design objectives. Considering consistency checking, a reasoner can verify whether or not a class can have any instances. An inconsistent class reflects something incorrectly built in the ontology. Both services have been extensively used when developing Ontoolcole, helping to assess that the formalization was correctly made. This ontology is thoroughly described below, although some low-level details are left out for the sake of readability.

As discussed in the precedent section, Ontoolcole aims to describe the capabilities of CSCL tools. This is accomplished by specifying that a `Tool` supports one or more `Tasks` performed by an `Actor`, maybe playing a specific `Role`. The realization of a task may require an `Artifact` as input or may produce an artifact as output. This simple schema, shown in Fig. 2(a), is the basis of the description of tools in Ontoolcole and it has been employed with small variations in the CSCL/CSCW literature [van der Veer and van Welie, 2000].

The `Actor` concept is specialized to `Person`, `Group` and `ComputerSystem`, representing the possible actors that can perform a task, as shown in Fig. 2(b). One first classification of tasks is determined by the actor that performs a task: an `IndividualTask` is performed by a person, a `GroupTask` is performed by a group and a `Computation` is performed by a computer system. Note that a group is composed of other groups and/or persons. In this sense, a group task denotes a collaborative task that can be performed either synchronously or asynchronously, reflecting an important characteristic of CSCL tools. This aspect is modeled

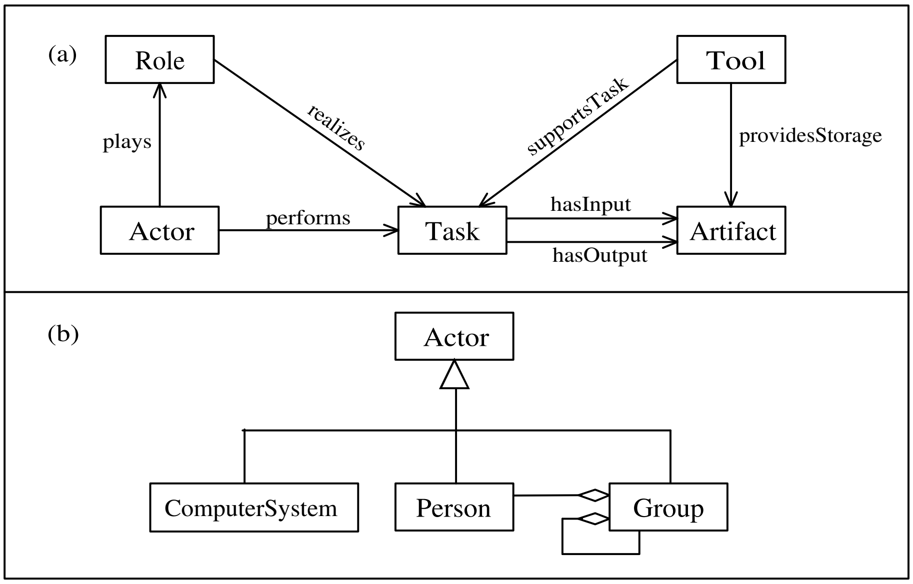

Figure 2: Overview of Ontoolcole: (a) Basic information model. (b) Actor types defined in Ontoolcole

in Ontoolcole using the property `hasTimeOfInteraction` that only applies to group tasks. Remarkably, the proposed information model can be related with the activity theory [Engëstrom, 2001]. This theory considers that individuals and groups learn something mediated by tools and other artifacts. Accordingly, the conceptual model of Ontoolcole represents those tools that can mediate in a learning activity and that can be used by persons or groups. The artifacts that are required as input or produced as output in Ontoolcole may represent the object which is the subject of study.

While the concepts `IndividualTask`, `GroupTask` and `Computation` reflect the actor performance, the nature of the task needs to be specified. Interestingly, [Koschmann, 1996] identifies the uses that may be served by a tool in a learning context: *presentation, support the creation of representational formalisms, computation, mediate communication and provide archival storage*. Since the information model of Ontoolcole represents tasks as perceived by actors, these terms have been reified to `Perception`, `Construction`, `Communication`, `Computation` (presented above) and `InformationManagement`. The key features of these five prototypical task types are shown in Table 3.1, indicating the possible values eventually assigned to the building blocks of the information model of Ontoolcole (see Fig. 2(a)). Note that these core task types can be further specialized to define other more specific types.

Task Type	**Actor**	**Role**	**Input**	**Output**
Perception	Person or Group	Beholder	Required	-
Construction	Person or Group	Editor	Optional	Required
Communication	Group	Communicator	-	-
Computation	Computer system	-	Required	Required
Inf. Management	Any	Publisher, Retriever...	Optional	Optional

Table 2: Key features of the five core tasks in Ontoolcole. These task types are inspired in [Koschmann, 1996] and represent the possible uses of a tool in a learning context

A perception task, such as reading or hearing, can only be performed by a person or a group and requires some artifact as input. Similarly, a construction task, such as writing or modeling, is performed by either a person or a group and produces some artifact as a result of the task. A communication task is explicitly performed by a group exchanging messages of different types (text, graphics, audio, video or documents). In contrast, an information management task can be specialized to publishing, retrieving, searching, sending or deleting some artifact. Depending on the type of information management task, an artifact would be required as input, e.g. publishing a document, or as output, e.g. retrieving an image. Finally, a computation task, such as compilation or computer simulation, is always performed by a computer system transforming some inputs into some outputs. Participants play roles during the realization of a task, as shown in Table 3.1. Roles are further specialized when appropriate, such as writer for a writing task (subclass of construction). As stated before, collaborative work is naturally described in this model by specifying group tasks and annotating them as either synchronous or asynchronous.

Using the precedent elements, other task types have been defined to obtain more refined concepts. A reasoner can then obtain the inferred hierarchy of task types, shown pictorially in Fig. 3. The root concept is `Task`, while the rest of task types are specializations of this concept. Note that concepts reflecting the actor performance and the nature of the task are intertwined in the resulting classification. In this sense, one of the top concepts is `InteractiveTask` which simply denotes any task performed by either a person or a group. Accordingly, `Perception` and `Construction` (as well as their subclasses) are specializations of `InteractiveTask`. They cannot be classified as `IndividualTask` because some instances of these concepts could be performed by a group; the same applies to `GroupTask` since some could be performed individually. On the other hand, `Communication` is classified as a `GroupTask` since it is always performed by a group.

Combining these elements it is possible to describe the capabilities of a tool

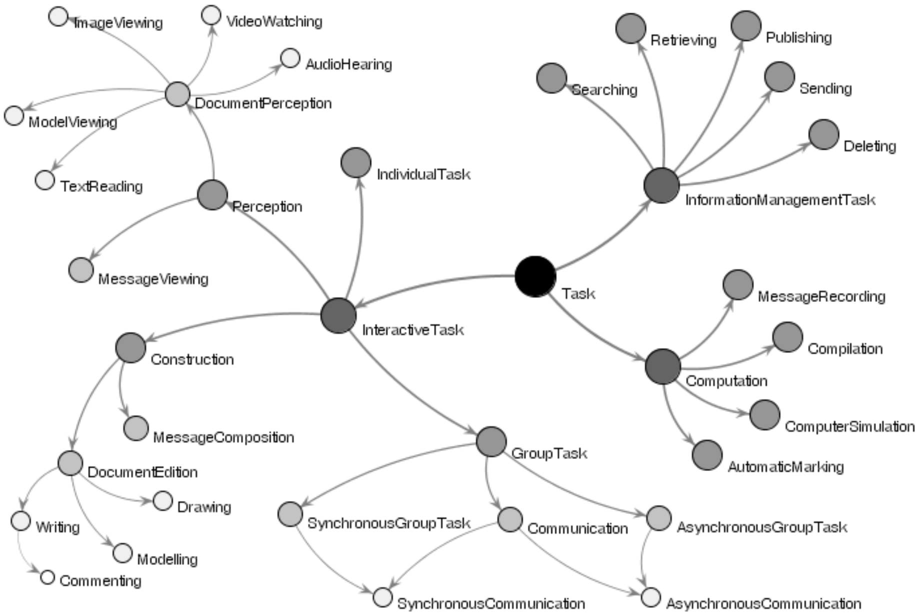

Figure 3: Inferred classification of task types defined in Ontoolcole. Concepts refering to the actor performance and the nature of the task dimensions are intertwined

by specifying the tasks that can be performed with it. Similarly, educators can use this model to pose queries expressing what is required to support their intended educational settings. In addition, Ontoolcole incorporates a predefined set of tool type definitions. These well-known tool types can be good starting points to perform a search. These tool definitions can be very generic, e.g. "communication tool: any tool that supports a communication task", or more specific, e.g. "whiteboard: any tool that allows a group to draw an image synchronously". Thus, during a search process, educators can directly ask for a tool type. Besides, they can also refine such a query with additional restrictions using the constructs of Ontoolcole. Fig. 4 shows an excerpt of the inferred classification of tool types. Note that the asserted tool concepts constitute a completely flat hierarchy before reasoning. Then, a reasoner classified them as shown in the figure using the definitions of these concepts.

The above description depicts Ontoolcole as in [Vega-Gorgojo et al., 2006]. However, this preliminary version has three main limitations that were detected when using it to annotate tool instances that could not be properly described. First of all, it lacks an **artifact model** to accommodate the different products that may be used or created during the realization of a task. This is a crit-

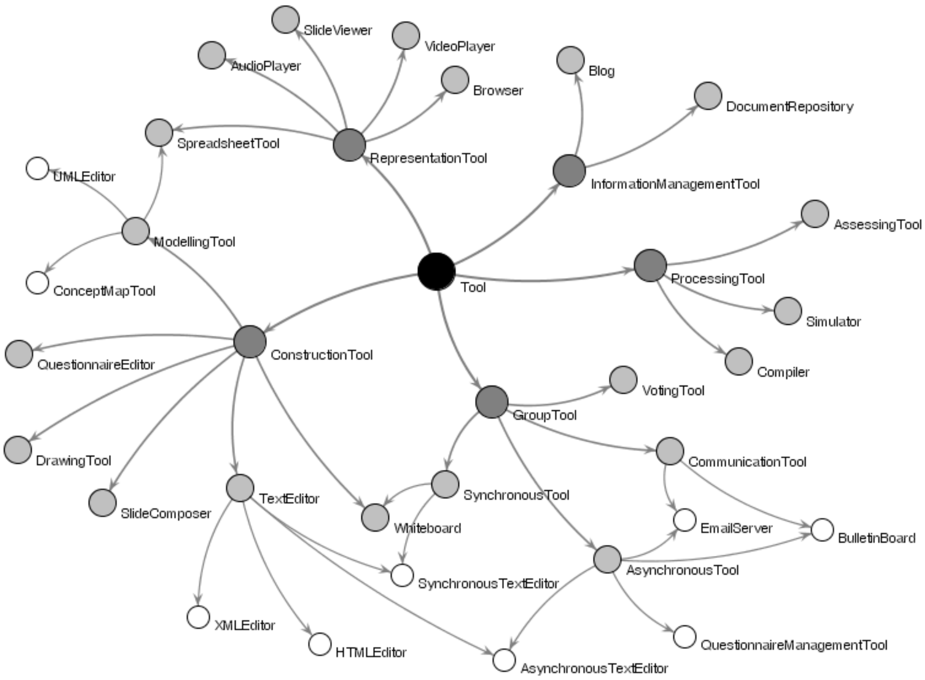

Figure 4: Excerpt of the inferred graph of tool types. Concepts were asserted in a completely flat hierarchy and the reasoner found the relationships shown in the figure

ical aspect to differentiate the capabilities of a tool and the previous version did not offer a mechanism for this issue. For instance, generic text editors like Word, XML editors like XMLSpy or specialized questionnaire editors like Adobe Designer were all described alike with the previous version of Ontoolcole, thus demanding an artifact model to differentiate among them.

Second, while Ontoolcole seems appropriate to describe tools designed for single tasks (for example a simple text editor), there are many CSCL tools that embody a complex workflow of tasks (for example a questionnaire management tool) that cannot be described with this simple schema. A **coordination model** is required for the description of the dynamic aspects of a tool.

Third, some tools such as document repositories or bulletin boards keep archival storage that can be accessed during different task realizations. Thus, some mechanism is required to describe the **static information resources** that a tool can manage.

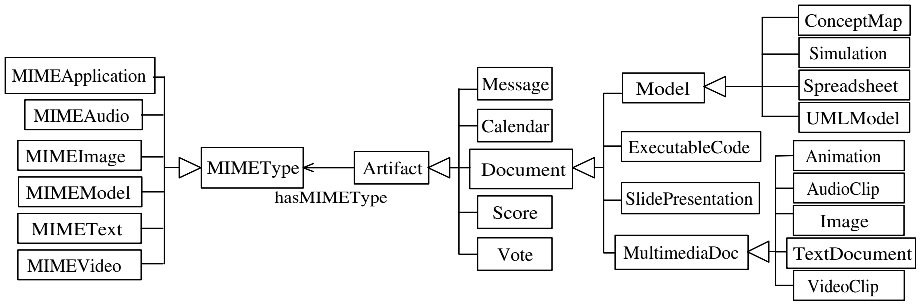

Figure 5: The new artifact model for Ontoolcole represented as a UML class diagram. Classes on the right depict artifact types while classes on the left the well-known MIME types. A specific artifact instance, e.g. a `TextDocument`, should be related to a MIME type instance, e.g. `application/pdf`

3.2 New Features of the Ontoolcole Ontology

As discussed above, an important limitation of the preliminary version of Ontoolcole was the **lack of an artifact model**, which hindered the description of tools and reduced the capability of tool discrimination. This new model should help to describe the artifacts required or produced during a task realization.

The resulting artifact model is shown pictorially in Fig. 5. The main design criterion has been to separate the artifact type from the format type. The well-known MIME types [IANA, 2006] have been employed to define the artifact format types. Interestingly, metadata standards such as LOM [IEEE, 2002] or Dublin Core Metadata Initiative (DCMI) [DCMI, 2006] also separate content from format. In contrast, our approach makes heavy use of inheritance to structure the artifact types, and explicitly defines a vocabulary that can be shared both by providers to describe tools and by educators to submit their queries, leveraging the search process. Moreover, this artifact model further describes some elements (although not shown in Fig. 5). For instance, a `TextDocument` incorporates a property to specify the type of text document (generic, questionnaire, source program, etc.).

This artifact model does not aim to represent all possible features of artifacts. This could be achieved including some of the metadata fields defined in LOM or DCMI. Though defining such information could be interesting for cataloging artifacts, in our case the intention is just to provide the required characteristics for the search of CSCL tools that were identified in subsection 2.2. Moreover, new artifact types can be easily incorporated to allow the description of a wider range of tools since we are aware that domain-specific tools, e.g. network simulators for teaching network protocols, will require extensions to this model.

In our previous work we also detected the **need to specify the resources a tool can store**. This feature is specially relevant for information management tools that keep archival storage of messages or documents. To solve this, a new property has been defined that relates tools with the artifacts they can manage. The name of this property is `providesStorage` and it is shown in Fig. 2(a). With this simple approach it is possible to specify that an e-mail server can store messages enclosing text and document files. Another example is the description of a persistent chat that can store the messages of a conversation.

Finally, the preliminary version **did not include a coordination model**. This is, perhaps, the most challenging limitation, and becomes quite important when describing asynchronous group scenarios, since different users may perform individual tasks that constitute a whole complex task. Furthermore, the utility of this feature is not limited to such cases; for instance, participants engaged in a synchronous group drawing task may exchange text messages at the same time to coordinate their drawing. Thus, some mechanism is necessary to describe the concurrent realization of the drawing and communication tasks.

Fig. 6(a) outlines the coordination model developed for Ontoolcole. Basically, a `CompositeTask` has been defined as any task that is composed of exactly one `ControlConstruct` element. This latter element acts as a container of tasks or other control constructs. Each control construct has different semantics: `Choice` implies that only one of the elements contained is performed; `Sequence` defines a time ordering; and `Split` serves to specify the concurrent execution of the elements contained. Combining these elements, it is possible to describe a wide range of complex workflows of tasks, e.g. a split of three concurrent sequences of tasks. Indeed, this coordination model is inspired in OWL-S Composite Processes [Martin et al., 2004a] that defines a mechanism for composing processes formalized in OWL DL. In contrast, the proposed model uses different constructs exploiting the high expressiveness of OWL DL. Specifically, `components` is defined as a transitive property and `composedOf` as a specialization of `components`; in this way, it is possible to obtain all the nested elements of a composite task simply asking a reasoner to retrieve the values of the property `components`.

An example of the use of this coordination model is shown in Fig. 6(b), representing a possible scenario of an asynchronous group writing task. This description provides a high-level overview of the group task (top of Fig. 6(b)) that is further refined with the description of the individual tasks performed by the members of the group (bottom of Fig. 6(b)). Remarkably, the sequence ordering is defined by the use of the `first` and `next` properties, pointing to the first and the subsequent tasks, respectively. Indeed, `first` is a specialization of `components` that is used to mark the first element in a sequence.

After overcoming the previous limitations, Ontoolcole has proven to be very versatile for the description of CSCL tools. Different instances of document

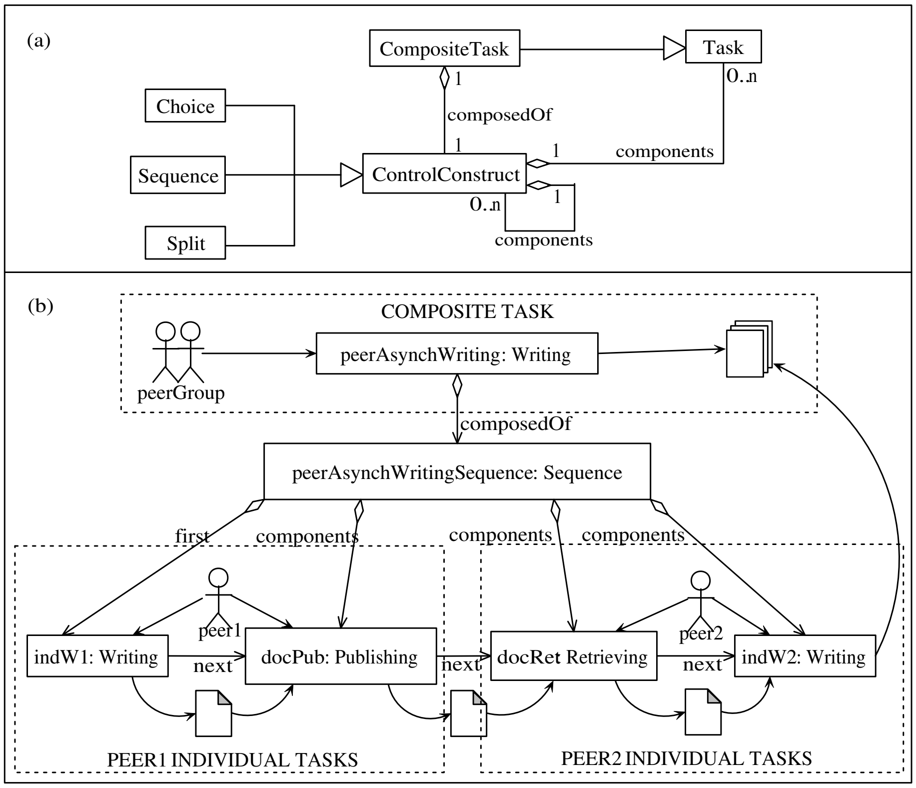

Figure 6: Ontoolcole's new coordination model (a). A `CompositeTask` is composed of a `ControlConstruct` element that includes other tasks or other control constructs. (b) illustrates the use of the coordination model for the description of an asynchronous group writing. A text document is produced as a result of a sequence of individual tasks performed by each of the members of the peer group. This figure only shows the properties that concern the coordination model for the sake of readability

repositories, drawing tools, whiteboards, collaborative text editors, multimedia players, questionnaire management tools, chats and concept map tools, among others, have been described with Ontoolcole. Exploiting Ontoolcole abstractions, tools are described by the depiction of their different scenarios of use. For instance, Fig. 6(b) forms the basis for the description of CoWeb [Rick et al., 2002], a popular asynchronous group text editor based on Wiki. Further aspects of CoWeb have been included such as the storage of text documents and images, document formats as well as the description of other CoWeb supported scenarios: individual text edition and retrieval of stored documents. Simpler tools can also be described in an easy way with Ontoolcole, such as the well-known pdf viewer Adobe Reader. This tool has been annotated by creating new instances of Tool (`adobeReader`), TextReading (`adobeReaderViewing`), Person (`adobeUser`) and TextDocument (`pdfDocument`) with MIME type `application/pdf`; in addition, it has been asserted that `adobeReader` supports the task `adobeReaderViewing` performed by `adobeUser` requiring as input `pdfDocument`.

A noteworthy consideration refers to the level of detail used to describe the capabilities of CSCL tools in Ontoolcole. An ontology should not contain all the possible information about the domain, but only the needed information for the intended applications [Noy and McGuinness, 2001]. Indeed, over-specification incurs in the cost of adding unnecessary complexity, making an ontology both difficult to maintain and exploit. In this sense, the design criterion in Ontoolcole has been to keep this ontology as simple as possible, as well as trying to simplify tool descriptions. For instance, CoWeb can support many different variations of group writing and the provided description corresponds to a simple one-step draft-passing process. However, it is sufficient to respond to the following queries: "a tool for asynchronous group writing"; "a tool that can store text documents"; "a tool that allows publishing text documents"; "a tool that allows retrieving text documents"; and any combination of the precedent ones. If we accept that educators will limit themselves to such queries, there are no extra benefits in providing descriptions of more complex and realistic scenarios of use. Indeed, it would be difficult for an educator to specify in a query his intended learning scenario. Moreover, the search facility would require significant complexity both to allow the specification of such queries and to match the tools that can support the scenario defined in the query. Thus, the proposed approach tries to simplify both tool descriptions and supported queries; whether this is sufficient or not will be discussed in section 5 which describes an evaluation experiment of Ontoolcole with real users.

A final remark can be made about tool annotation, since it requires a deep understanding of Ontoolcole in order to provide comprehensive descriptions. Furthermore, the popular ontology editor Protégé [Knublauch et al., 2004] has been employed for this issue and using Protégé needs some training for non-trivial usage. Thus, ongoing efforts should be carried out to develop an Ontoolcole-flavoured authoring system to allow providers to describe their own tools. The issue of tool annotation could be mitigated with tool templates, allowing providers to adapt an appropriate template to a specific tool instance. Templates could be based on the definitions of tool types provided in Ontoolcole. For instance, a provider could select the template of a whiteboard (defined previously) and adapt it afterwards, e.g. indicating supported MIME type of the images produced with the whiteboard.

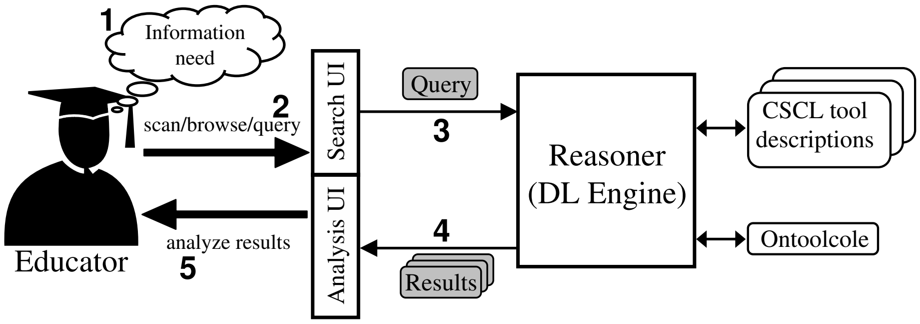

Figure 7: Architecture of Ontoolsearch and user interactions. (1) Initially, the educator has a preliminary idea of the CSCL tool he requires. (2) Then, he uses the search UI to scan the information structure, to browse tool types and to formulate queries. (3) Queries are forwarded to the reasoner. (4) The reasoner manages the knowledge base composed of Ontoolcole and the descriptions of CSCL tools; it responds to a query with the set of tool instances that match the query. (5) These results are presented to the educator through the analysis UI, allowing him to evaluate their appropriateness

4 Ontoolsearch: a System for the Search of CSCL Tools

Section 3 depicted Ontoolcole and illustrated how to use it to semantically annotate CSCL tools. However, in order to fully support educators in the search of CSCL tools we are currently developing a recommender system of CSCL tools that uses Ontoolcole in the back-end to structure tool metadata. This system is named Ontoolsearch and it is described here to illustrate the design of the target application of the proposed ontology.

Ontoolsearch is an interactive system intended for educators. Since information seeking is an imprecise process, when users approach an information access system they often have only a vague understanding of how they can achieve their goals [Hearst, 1999]. Thus, Ontoolsearch should help educators in the process of understanding and expression of their information needs. Ontoolsearch supports two main activities: search for tools and analysis of results. To perform the former activity, educators use Ontoolsearch to scan the information structure defined in Ontoolcole, to browse the category hierarchy of tool types or to formulate a query. Retrieved tools are presented to educators, allowing them to analyze the results. These user interactions are shown in Fig. 7 together with the architecture of Ontoolsearch.

Queries are modeled as OWL DL concepts and are processed by a reasoner. The reasoner offers an instance retrieval service that is used to obtain the results of a query. Current prototype of Ontoolsearch employs the reasoner RacerPro [Racer, 2007] for this issue, managing the knowledge base composed of Ontoolcole (the structure) and the descriptions of CSCL tools (the data). A query is formed using the abstractions defined in Ontoolcole (see section 3). Thus, it is possible to ask for tool types, task types, actors that can perform a task, required artifacts, elements in a composite task, etc. A query expression can be articulated using the language nRQL [Racer, 2007] and submitted to RacerPro. A simple example of an nRQL query is shown below:

```
(retrieve (?TOOL)
  (and (?TOOL |Tool|)(?TASK |Communication|)(?MESSAGE |Message|)
    (?TASK (STRING= |hasTimeOfInteraction| "synchronous"))
    (?MESSAGE (STRING= |encloses| "text"))
    (?TASK ?MESSAGE |allowsMessage|)<
    (?TOOL ?TASK |supportsTask|))
```

This query asks for tool instances that match the restrictions defined in the `AND` clause: supporting a synchronous communication task based on text messages. However, although nRQL is a very powerful and flexible command language, its syntax is complex and we cannot assume that educators will learn nRQL. This leads to an important trade-off in all user interface designs: simplicity versus power [Hearst, 1999]. Moreover, it is expected that educators will search CSCL tools intermittently, pushing to a simpler and easier to learn interface. Thus, the decision in Ontoolsearch has been to offer a graphical direct manipulation interface at the expense of less flexibility than nRQL, limiting the expressiveness of queries. Direct manipulation interfaces are commonly easier to use than other methods [Shneiderman, 1997, p. 205], due to: (1) continuous representation of the object of interest, (2) physical actions or button presses instead of complex syntax, and (3) rapid incremental reversible operations whose impact on the object of interest is immediately visible. The search interface captures user interactions to form a query that will be transformed to an nRQL expression to be processed by RacerPro.

In order to design the search interface of Ontoolsearch, we have followed a participatory design strategy involving both software developers and educators. Participatory design is a common methodology to involve stakeholders to identify requirements and give feedback of preliminary prototypes [Muller and Kuhn, 1993]. Thus, several search user interfaces have been iteratively proposed until having agreed the final interface that is shown in Fig. 8. It has been designed following some well-known principles for information access [Hearst, 1999]. First, the upper left frame displays the inferred graph of CSCL tool types (see graph frame in Fig. 4) and is intended to provide educators with a *good starting point* for a search. Educators can browse tool categories and eventually select a tool

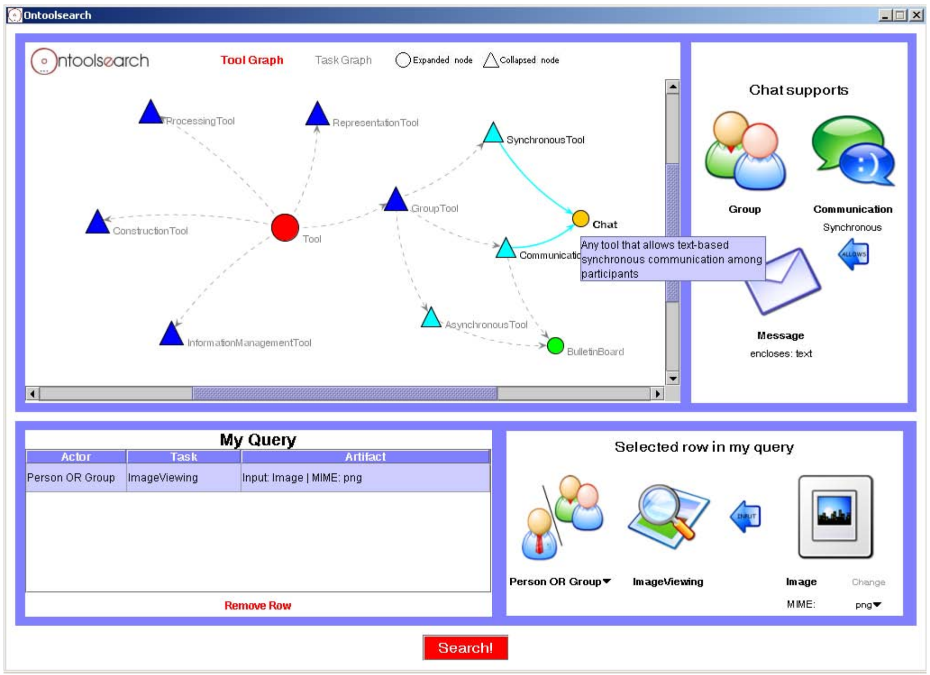

Figure 8: Snapshot of the search user interface of Ontoolsearch

type and add it to the query frame (bottom-left in Fig. 8) as a list of one or more actor-task-artifact rows, depending on the specific tool type definition; then, pressing the 'Search' button will retrieve the set of tool instances pertaining to that category. Educators can also manipulate the graph by moving, expanding, collapsing and hiding nodes, allowing them to control the information to display. This graph has been developed using the JUNG Framework [O'Madadhain et al., 2003], a flexible and open-source Java library for graph visualization.

Second, since humans are highly attuned to images and visual information, the upper right frame (visualization frame) displays a *visual representation* of the currently selected node of the graph. This is accomplished using icons to represent the tasks supported by a tool type, along with the actors and artifacts, as shown in Fig. 8. This aspect allows rapid communication of the abstractions modeled in Ontoolcole. Besides, tooltips are enabled in the graph, showing textual descriptions stored in Ontoolcole when the mouse is positioned over a node. Both the visualization frame and the tooltips allow educators to scan the information structure of Ontoolcole.

Third, Ontoolsearch seamlessly *integrates scanning, browsing and querying* in the search interface. Precedent paragraphs depicted the integration of tool browsing with scanning information structure. Browsing tool categories is a way of providing a set of predefined queries that may satisfy the information needs of educators. However, they need more control to formulate a query, specifying additional restrictions concerning tasks, actors and artifacts. To accomplish it, each actor-task-artifact row in the query frame can be selected, triggering its visualization in the lower right frame (editable frame); then, educators can add constraints manipulating the elements of the graphical interface, e.g. pressing a button to include a task type, restricting the actor in a combo or selecting a MIME type in a pop-up window. In addition, educators can switch the graph frame to display the task category hierarchy (see Fig. 3), allowing them to directly add a task type to the query frame. This way, we envision educators will initially browse the tool category graph and retrieve a set of tool instances pertaining to a tool category; then, as they familiarize with the information structure, they will eventually switch to the task category graph and begin to refine their queries.

For the sake of simplicity, the search interface limits the expression of some features in a query that could otherwise be specified in nRQL. Specifically, educators cannot ask for the artifact types that can be stored by a tool. However, this feature is implicitly included in a query when an information management task is added to the query frame, e.g. an educator may require a tool that allows publishing messages and the query expression will incorporate the capability of storing messages, although this will not be shown to the educator to reduce his working memory load. In addition, educators cannot specify the components of a composite task in a query, although Ontoolsearch will retrieve any tool that matches the list of tasks contained in the query frame even in the case that some of them form part of a composite task. Finally, Ontoolsearch constructs queries using an AND clause to include the elements in the query frame; although a combination of AND and OR clauses is possible, most users have great difficulty in specifying Boolean queries [Greene et al., 1990].

Concerning the analysis user interface, we have followed the design principle of *placing the results set in the context of other information types*. Specifically, tool categories are used to place the results, constructing a graph showing the tool instances as well as the tool types to which they belong. Selecting an instance shows a combination of textual and graphical information of its Ontoolcole-compliant description. An educator can then go back to refine his query, start a new one or access the session history to restore a previous query and its results. During this process, he will eventually find some tools that satisfy his information needs.

The description so far shows how Ontoolsearch could be used by educators to search appropriate CSCL tools for their educational settings. Besides, it can be extended to support the discovery of CSCL services. As discussed in section 2, a service description comprises many attributes, although service discovery basically requires the service capabilities. The capabilities of CSCL services can be described with Ontoolcole, enabling the use of Ontoolsearch for the discovery of CSCL services. This can be accomplished in cooperation with a service registry, such as UDDI. The registry stores service records with the necessary information to invoke a service. This way, each Ontoolcole tool description will point to the set of CSCL services that implement the tool functionality. Thus, Ontoolsearch could be extended with a module that maps tool descriptions to service implementations for the search of CSCL services. Then, educators could search appropriate services using Ontoolsearch. After selecting a service, it would be integrated in the CLS using the binding information stored in the registry. Current service-oriented CLSs like Gridcole [Bote-Lorenzo et al., 2007b] offer the required mechanisms to seamlessly perform the integration of a CSCL service once it has been selected. Noteworthy, other works such as Feta [Lord et al., 2005] and Matchmaker [Kawamura et al., 2005] propose similar approaches to enhance UDDI's search functionalities for specific domains.

Finally, service descriptions could incorporate quality of service parameters to define important objective, e.g. cost, availability, request-to-response time, and subjective attributes, e.g. user ratings. This will allow the expression of additional restrictions in a query that are specially suited for service selection, once discovered services with desired capabilities. For example, [Maximilien and Singh, 2004] proposes an ontology for the description of quality of service parameters that could be integrated in Ontoolsearch for the issue of service selection.

5 Ontoolcole evaluation

Sections 3 and 4 portrayed Ontoolcole and Ontoolsearch, respectively. Since they have been designed with the aim of facilitating educators the search of CSCL tools, it is necessary to evaluate if they accomplish this goal. At the moment of the evaluation Ontoolsearch was not available, so the evaluation experiment presented here focuses on Ontoolcole. Specifically, we want to assess whether educators' real questions for services can be formulated with Ontoolcole. Besides, using a small testbed of 21 CSCL tools like chats or document repositories, we will determine if the tools retrieved satisfy users' expectations for the questions they posed. To pursue these objectives, we have devised a case study engaging several educators to pose non-restricted questions from their real practice.

Some methodologies for ontology evaluation such as [Grüninger and Fox, 1995] propose the use of competency questions defined *a priori* and tested afterward against an ontology to evaluate its compliance. In contrast, the proposed experiment involves real users to express their information needs based on their practice and without any restriction on the questions posed. While the goal of the former methodology is to evaluate whether an ontology can respond to a subset of predefined questions, the aim of the devised experiment is to assess whether a specific ontology can fit the information needs of a community of users.

5.1 The experimental setup

In order to evaluate whether educators can search CSCL tools using Ontoolcole we have devised a Wizard of Oz experiment [Kelley, 1984]. Since Ontoolsearch was not available at the time of the experiment, this methodology allows users to interact with a system missing some functionalities. Specifically, Ontoolsearch's user interface was not available although it was possible to submit nRQL queries to RacerPro fed with Ontoolcole and a testbed of Ontoolcole-compliant tool descriptions. This way, 15 education practitioners were recruited and demanded to formulate questions for tools.

The instructions given to the educators were to briefly describe a learning setting based on their real practice that requires software tools to be enacted. Then, they had to pose questions in natural language using unlimited expressiveness in their formulation to find appropriate tools for their defined settings. It was suggested to provide the description of three different learning settings and one question for each one. Most of the educators submitted three settings/questions, and some of them defined a unique learning scenario and posed various questions about it. A human mediator played the "wizard" and was in charge of translating user questions into nRQL queries. Translated queries were paraphrased back to natural text and returned to participants in addition to the tools found by RacerPro. Thus, the "wizard" replaced the missing human-computer interface, enabling the evaluation of the initial goals of the experiment. He did not ask for clarifications in order to not affect the results of the study. Finally, users provided feedback about the translation quality and the utility of the tools found for their educational settings.

An overview of the profiles of the 15 participants in the experiment is shown in Table 5.1. Most of them are professors or teachers either in Computer Science, Telematics, Signal Theory or Pedagogy. Remarkably, 9 out of 15 are practitioners in CSCL and 7 have some knowledge about Ontoolcole. It was expected that CSCL practitioners posed questions about CSCL settings, although they were not obliged to. Besides, evaluation results might be influenced by prior users' knowledge about Ontoolcole. Section 5.2 further discusses these aspects.

In addition, a reasonable amount of educational tools were described using Ontoolcole, in order to validate the retrieval part. For the experiment we prepared 21 tool descriptions, including text editors, browsers, document visualizers, authoring tools, document repositories, questionnaire management tools,

Faculty Position	# of Users	Area of Knowledge				CSCL Practitioner		Ontoolcole Knowledge		
		CS	TM	ST	PG	Yes	No	Yes	Fair	No
Professor	5	-	3	-	2	5	-	2	1	2
Teacher	8	2	3	1	2	4	4	2	1	5
Grant holder	1	1	-	-	-	-	1	-	-	1
Student	1	-	1	-	-	-	1	1	-	-
Total	15	3	7	1	4	9	6	5	2	8

Table 3: User profiles of the participants in the experiment. 'CS' stands for 'Computer Science', 'TM' for 'Telematics', 'ST' for 'Signal Theory' and 'PG' for 'Pedagogy'

Degree Studies	# of Questions	Setting			CSCL	
		Face-to-face	Distant	Blended	Yes	No
MsC Computer Science	8	3	1	4	3	5
MsC Telecommunications	20	13	4	3	12	8
PhD Telecommunications	7	1	3	3	6	1
MsC Pedagogy	5	1	1	3	3	2
Total	40	18	9	13	24	16

Table 4: Overview of the context of the analyzed questions

chats, e-mail, drawing tools, spreadsheets and concept map tools. Most of these tools are generic and have been employed in some courses following a CSCL methodology in our University. Thus, we expected they could be useful for questions referred to CSCL settings. However, the used dataset is too small to find adequate tools for very specific needs. In this sense, a bigger dataset would be valuable for a deeper analysis of the query results. This is part of future work.

5.2 Evaluation results

The participants in the experiment formulated 40 questions. An overview of the context of these questions is shown in Table 4. They concern courses on Computer Science, Telecommunications and Education as well as a doctoral course on Research Methodology. Besides, questions present a balanced mixture of face-to-face, distant and blended settings, while 24 out of 40 questions were conceived for the practice of CSCL. They refer to different activities such as UML modeling, signal/noise calculations, network simulations, design of didactic units, HTML authoring, group writing or document sharing among others.

The 40 questions were translated into nRQL queries and submitted to RacerPro. With the translated queries and the retrieved tools, a personalized response was sent to the participants of the experiment. Users' feedback was collected concerning the quality of the translated queries and the usefulness of found tools.

Id	Educator question	Translation
Q1	*I need a tool for visualizing PDF documents*	`any tool that supports the perception of documents with MIME type PDF`
Q2	*I want a collaborative concept map tool*	`any tool that supports the modeling of concept maps in group`
Q3	*I'm searching a tool that supports working with your companion, although not only working, but also requiring sharing information between them, not just dividing the work in two parts. Besides, it should allow seeing what your companion makes and support communicating with her in an easy way*	`any tool that supports the construction of artifacts in group and, besides, supports the communication among group members`
Q4	*I want a tool that can facilitate a meeting among co-located and remote groups*	`any tool that supports the synchronous communication among group members`
Q5	*I need a network simulator that includes the TCP protocol*	`any simulator`

Table 5: Sample educator questions and translations

Five of the user questions were very open and the wizard proposed several alternative queries to fit each original question. All the alternative queries were submitted to the educator and he judged whether they satisfied his information needs or not. The educator then provided a single evaluation for the whole set of alternatives, neither for the best nor the worst of them.

Table 5.2 shows five sample questions posed by educators, as well as the translations to nRQL paraphrased to natural language. Some questions, such as Q1 and Q2, used similar abstractions as those supported by Ontoolcole and were easily translated to nRQL without any information loss. However, in other cases questions were more abstractly defined and used different concepts as those modeled in Ontoolcole. For example, questions Q3 and Q4 required significant effort in the translation. Hence some of the features required by educators could not be expressed with Ontoolcole, such as the awareness part of question Q3.

The quantitative results of the study are summarized in table 5.2. Figures under column 'Translation Understanding' show that most participants understood very well the translated queries, indicating that Ontoolcole's abstractions are comprehensible. Significantly, users ranked with an overall 3.64 (from 0 to 5) the quality of the translations. Taking into account that users were not restricted on the questions they posed, this figure can be considered very positive. In this sense, some users formulated questions at an upper level than that supported by Ontoolcole, using high-level abstractions such as *group memory* and *presential debates*. In contrast, these were translated to `stored documents that can`

Question Set		Translation Understanding	Quality of the Translation	Found Tools (%)	Tool Utility
CSCL	24	4.88	3.85	87%	4.32
Non-CSCL	16	4.71	3.29	75%	2.21
Ontoolcole-aware: Yes	16	4.94	3.28	100%	3.02
Ontoolcole-aware: Fair	10	4.70	3.40	70%	4.43
Ontoolcole-aware: No	14	4.75	4.33	71%	3.80
Total	40	4.82	3.64	82%	3.55

Table 6: Evaluation results summarized using standard means. Levels are: 5 (very good), 4 (good), 3 (somewhat good), 2 (somewhat bad), 1 (bad) and 0 (very bad). The first two rows distinguish questions referred or not to CSCL settings. Following three rows present the evaluation results according to the users' knowledge of Ontoolcole. Final row shows the aggregated figures

`be accessed by a group` and `synchronous communication`. Users were aware of this 'semantic gap' that implied a simplification in the translations, although they considered that the essence of the questions was preserved in the queries. In other cases there were some misunderstandings due to natural language ambiguity. For instance, the "wizard" interpreted that a user demanded a programming environment when she required a tool for program designing (possibly a UML modeling application). Curiously, users aware of Ontoolcole were more demanding than the rest (3.28 vs 4.33), although the "wizard" feels that some of the questions were thought as a challenge to Ontoolcole.

Interestingly, questions referred to CSCL settings were ranked a 17% better than non-CSCL. Non-CSCL questions were sometimes very domain-specific demanding specialized tools such as network simulators, specialized computer graphics applications or very specific authoring tools for psycho-pedagogy, just to mention a few. For instance, the translation of question Q5 in table 5.2 was ranked with level '1' (bad). Moreover, most of the tools in the dataset are not domain-specific but collaborative tools to support communication and group work that can be useful in a wide range of CSCL settings, affecting the utility figure. This fact explains that educators considered useful the majority of tools found for CSCL questions, such as CmapTools for question Q2. On the contrary, questions referred to non-CSCL settings matched no instance, or rather generic ones that were ranked poorly (2.21) by educators. However, the core of Ontoolcole (tasks, artifacts, etc.) fits these queries but extensions concerning specific artifact types (e.g. data-network model) would increase the precision of responses. In this sense, specialized extensions for Ontoolcole seem a feasible approach to exploit Ontoolcole in specific domains.

To conclude the analysis, users suggested some enhancements to leverage

Ontoolcole. Excluding specialized domain-extensions, CSCL users demanded the description of awareness, advanced commenting and meeting capabilities. We are currently considering the inclusion of these suggested new features in Ontoolcole. Other users required the inclusion of technological features in their queries such as quality of service parameters, discussed in section 4. Further comments can be made about query expressions. Using the Ontoolcole model, most of educators' questions could be easily formulated. It seems that the abstractions employed in Ontoolcole, namely tool types, task types, artifact types and actors, can serve for the expression of their queries. The artifact model was employed in 67% of the queries, indicating that this extension was necessary. Remarkably, more complex queries involving composite tasks and the storage of artifacts were only used in 16% and 8% of the queries, respectively. This fact indicates that the approach followed to design Ontoolsearch can be effective taking into account the simplicity/power tradeoff. A new experiment using Ontoolsearch will eventually show evidence of this issue.

6 Conclusions and Future Work

Supporting educators in the semantic search of CSCL services is a challenging task. This paper presented significant contributions toward this goal. Briefly, an enhanced version of Ontoolcole, an ontology to semantically annotate CSCL tools; the evaluation of this ontology with real users involved in a Wizard of Oz experiment; and Ontoolsearch, a graphical interactive facility for the search of CSCL tools.

Ontoolcole defines the structure of the information required to describe the capabilities of CSCL services. The abstractions modeled in Ontoolcole intend to be meaningful for educators, allowing them to search CSCL services. This revised version incorporates a set of new features that allows a much more complete and precise description of CSCL tools. Namely, the artifact model developed defines an extensible structured depiction of the artifact types that can be employed during the execution of CSCL activities. The formatting information of the artifacts is described using the well-known MIME types. In addition, a mechanism has been built for specifying the resources a tool can store. Finally, a coordination model has been incorporated for the description of complex workflows of tasks.

In addition, we carried out a case study with real users to evaluate whether Ontoolcole can be employed by educators to search CSCL tools. 15 participants expressed their information needs for tools in the context of their real practice without any restriction on the questions they posed. A "wizard" translated those questions into queries and submitted them to an Ontoolcole-powered retrieval system. Participants ranked with an overall 3.64 (from 0 to 5) the quality of the translations, indicating that Ontoolcole has the semantic richness to support these queries. Significantly, questions referred to CSCL settings were ranked a 17% better due to the emphasis of Ontoolcole on collaboration features. Moreover, the tools retrieved were rated with 4.32 (from 0 to 5) for the CSCL case, evidencing that users found useful tools for their educational needs. Real questions proposed by education practitioners are difficult to translate to SWS languages like OWL-S, since they employ low-level service abstractions that do not easily match educators' intended questions. Interestingly, this problem has been also reported for user oriented semantic service discovery in the very different domain of bioinformatics [Lord et al., 2005].

To make Ontoolcole usable by practitioners, we have devised Ontoolsearch, an interactive system for the search of CSCL tools that uses Ontoolcole in the back-end to structure tool metadata. Ontoolsearch offers an easy to use direct manipulation interface. It seamlessly integrates scanning, browsing and querying in the search interface and provides visual representations of the underlying ontology. Furthermore, Ontoolsearch could be extended to support the discovery of CSCL services. It could be accomplished in coordination with a service registry and a mapping module that links an Ontoolcole-compliant tool description to a set of services that implement the tool functionality.

Currently, we are completing a prototype of Ontoolsearch that will be evaluated during the first half of 2007. Ontoolsearch aims at substituting the "wizard" in the precedent experiment, allowing educators to autonomously search CSCL services. It is expected that with such a system, educators would be able to translate their information needs into appropriate queries, even if this process takes them several tries. Then, a new experiment will be performed in order to assess whether educators can accomplish a complete tool search scenario. Besides, we are considering users' suggestions to produce a new version of Ontoolcole including new features such as awareness or course management functionalities. Additionally, we plan the integration of Ontoolcole/Ontoolsearch in the service-oriented, tailorable CLS Gridcole [Bote-Lorenzo et al., 2007b] to support the discovery of CSCL services.

Acknowledgements

This work has been funded by the European Commission funded Kaleidoscope network of excellence FP6-2002-IST-507838, Spanish Ministry of Education and Science project TSI2005-08225-C07-04 and Autonomous Government of Castilla and León, Spain, project VA009A05.

References

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley, Harlow, UK, first edition.

[Baker and Lund, 1996] Baker, M. and Lund, K. (1996). Flexibly structuring the interaction in a CSCL environment. In *Proceedings of the European Conference on Artificial Intelligence in Education*, pages 401–407, Lisbon, Portugal.

[Bechhofer et al., 2004] Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P., and Stein, L. (2004). OWL web ontology language reference. Recommendation, W3C. URL: http://www.w3.org/TR/owl-ref/, last visited May 2007.

[Betbeder, 2003] Betbeder, M. L. (2003). *Symba: a Tailorable Framework to Support Collective Activities in a Learning Context.* PhD thesis, Université du Maine, France. In French.

[Bote-Lorenzo et al., 2007a] Bote-Lorenzo, M. L., Asensio-Pérez, J. I., Gómez-Sánchez, E., Vega-Gorgojo, G., Dimitriadis, Y. A., and Guíñez-Molinos, S. (2007a). A grid service-based collaborative network simulation environment for computer networks education. In *Proceedings of the Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007)*, Niigata, Japan.

[Bote-Lorenzo et al., 2007b] Bote-Lorenzo, M. L., Gómez-Sánchez, E., Vega-Gorgojo, G., Dimitriadis, Y. A., Asensio-Pérez, J. I., and Jorrín-Abellán, I. M. (2007b). Gridcole: a tailorable grid service based system that supports scripted collaborative learning. *Computers & Education.* Accepted for publication.

[Bote-Lorenzo et al., 2004] Bote-Lorenzo, M. L., Hernández-Leo, D., Dimitriadis, Y. A., Asensio-Pérez, J. I., Gómez-Sánchez, E., Vega-Gorgojo, G., and Vaquero-González, L. M. (2004). Towards reusability and tailorability in collaborative learning systems using IMS-LD and grid services. *Advanced Technology for Learning*, 1(3):129–138.

[Curbera et al., 2002] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., and Weerawarana, S. (2002). Unraveling the Web Services Web. *IEEE Internet Computing*, 6(2):86–93.

[DCMI, 2006] DCMI Usage Board (2006). DCMI metadata terms. Specification 1.1, DCMI. URL: http://dublincore.org/documents/dcmi-terms/, last visited April 2007.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

[Engëstrom, 2001] Engëstrom, Y. (2001). Expansive learning at work: toward an activity theoretical reconceptualization. *Journal of Education and Work*, 14(1):133–156.

[Fensel, 2004] Fensel, D. (2004). *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce.* Springer, second edition.

[Friesen and Mazloumi, 2004] Friesen, K. and Mazloumi, N. (2004). Integration of learning management systems and web applications using web services. *Advanced Technology for Learning*, 1(1):16–24.

[Greene et al., 1990] Greene, S. L., Devlin, S. J., Cannata, P. E., and Gómez, L. M. (1990). No IFs, ANDs, or ORs: A study of database querying. *International Journal of Man-Machine Studies*, 32(3):303–326.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 6(2):199–221.

[Grüninger and Fox, 1995] Grüninger, M. and Fox, M. S. (1995). Methodology for the design and evaluation of ontologies. In Skuce, D., editor, *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI-95)*, Montreal, Canada.

[Guha et al., 2003] Guha, R., McCook, R., and Miller, E. (2003). Semantic search. In *Proceedings of the Twelfth International World Wide Web Conference (WWW2003)*, Budapest, Hungary.

[Hearst, 1999] Hearst, M. A. (1999). User interfaces and visualization. In [Baeza-Yates and Ribeiro-Neto, 1999], pages 257–323.

[Hull et al., 2006] Hull, D., Zolin, E., Bovykin, A., Horrocks, I., Sattler, U., and Stevens, R. (2006). Deciding semantic matching of stateless services. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI 2006)*, Boston, MA, USA.

[IEEE, 2002] IEEE Learning Technology Standards Committee (2002). IEEE standard for learning object metadata. Specification 1484.12.1-2002, Computer Society/Learning Technology Standards Committee.

[IMS, 2003] IMS Global Learning Consortium (2003). IMS learning design information model. Specification, IMS.

[IMS, 2006] IMS Global Learning Consortium (2006). IMS Tools Interoperability Guidelines. Specification 1.0, IMS.

[IANA, 2006] Internet Assigned Numbers Authority (IANA) (2006). MIME media types. URL: http://www.iana.org/assignments/media-types/, last visited May 2007.

[Jonassen, 2006] Jonassen, D. H. (2006). *Modeling with Technology. Mindtools for conceptual change.* Pearson Education, Inc., Upper Saddle River, NJ, USA, third edition.

[Kawamura et al., 2005] Kawamura, T., Hasegawa, T., Ohsuga, A., Paolucci, M., and Sycara, K. (2005). Web services lookup: a matchmaker experiment. *IT Professional*, 7(2):36–41.

[Kelley, 1984] Kelley, J. F. (1984). An iterative design methodology for user-friendly natural language information applications. *ACM Transactions on Office Information Systems*, 2(1):26–41.

[Klein and Bernstein, 2004] Klein, M. and Bernstein, A. (2004). Toward high-precision service retrieval. *IEEE Internet Computing*, 8(1):30–36.

[Knublauch et al., 2004] Knublauch, H., Fergerson, R. W., Noy, N. F., and Musen, M. A. (2004). The Protégé OWL plugin: An open development environment for semantic web applications. In *Proceedings of the Third International Semantic Web Conference (ISWC 2004)*, LNCS 3298, pages 229–243, Hiroshima, Japan.

[Koschmann, 1996] Koschmann, T. (1996). Paradigm shift and instructional technology. In Koschmann, T., editor, *CSCL: Theory and Practice of an emerging paradigm*, pages 1–23. Lawrence Erlbaum, Mahwah, NJ, USA.

[Lara et al., 2004] Lara, R., Roman, D., Polleres, A., and Fensel, D. (2004). A Conceptual Comparison of WSMO and OWL-S. In *Proceedings of the European Conference on Web Services (ECOWS 2004)*, LNCS 3250, pages 254–269, Erfurt, Germany. Springer.

[Lassila, 1998] Lassila, O. (1998). Web metadata: A matter of semantics. *IEEE Internet Computing*, 2(4):30–37.

[Lord et al., 2005] Lord, P., Alper, P., Wroe, C., and Goble, C. (2005). Feta: A lightweight architecture for user oriented semantic service discovery. In *Proceedings of the Second European Semantic Web Conference (ESWC 2005)*, LNCS 3532, pages 17–31, Heraklion, Greece. Springer.

[Martin et al., 2004a] Martin, D., Burstein, M., Hobbs, J., Lassila, O., et al. (2004a). OWL-S: Semantic markup for web services. White paper OWL-S 1.1, DARPA Agent Markup Language Program. URL: http://www.daml.org/services/owl-s/1.1/overview/, last visited April 2007.

[Martin et al., 2004b] Martin, D. et al. (2004b). Bringing semantics to web services: The OWL-S approach. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, San Diego, CA, USA.

[Maximilien and Singh, 2004] Maximilien, E. M. and Singh, M. P. (2004). A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8(5):84–93.

[Morch, 1995] Morch, A. (1995). Three levels of end-user tailoring: customization, integration and extension. In *Proceedings of the Third Decennial Aarhus Conference*, pages 41–45, Aarhus, Denmark.

[Muller and Kuhn, 1993] Muller, M. J. and Kuhn, S. (1993). Participatory design. *Communications of the ACM*, 36(6):24–28.

[Noy and McGuinness, 2001] Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Knowledge Systems Laboratory.

[O'Madadhain et al., 2003] O'Madadhain, J., Fisher, D., White, S., and Boey, Y. (2003). The JUNG (java universal network/graph) framework. Technical Report UCI-ICS 03-17, University of California, Irvine, CA, USA.

[OASIS, 2004] Organization for the Advancement of Structured Information Standards (OASIS) (2004). Universal Description, Discovery and Integration v3.0.2 (UDDI). Specification 3.0.2, OASIS. URL: http://uddi.org/pubs/uddi_v3.htm, last visited April 2007.

[Paolucci et al., 2002] Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. (2002). Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference (ISWC 2002)*, pages 333–347, Sardinia, Italy. Springer.

[Paolucci and Sycara, 2003] Paolucci, M. and Sycara, K. (2003). Autonomous semantic web services. *IEEE Internet Computing*, 7(5):34–41.

[Papazoglou, 2003] Papazoglou, M. P. (2003). Service-oriented computing: concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE 2003)*, pages 3–13, Roma, Italy.

[Papazoglou, 2007] Papazoglou, M. P. (2007). Web services technologies and standards. *Computing Surveys*. To appear in 2007.

[Papazoglou and Georgakopoulos, 2003] Papazoglou, M. P. and Georgakopoulos, D. (2003). Service-oriented computing. *Communications of the ACM*, 46(10):25–28.

[Racer, 2007] Racer Systems GmbH & Co.KG (2007). RacerPro website. URL: http://www.racer-systems.com/products/racerpro/index.phtml, last visited May 2007.

[Rick et al., 2002] Rick, J., Guzdial, M., Carroll, K., Hollaway-Attaway, L., and Walker, B. (2002). Collaborative learning at low cost: CoWeb use in english composition. In Stahl, G., editor, *Proceedings of the Computer Supported Collaborative Learning Conference 2002 (CSCL 2002)*, Boulder, CO, USA. Lawrence Erlbaum Associates.

[Roman et al., 2005] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106.

[Roschelle et al., 1999] Roschelle, J., DiGiano, C., Koutis, M., Repenning, A., Philips, J., Jackiw, N., and Suthers, D. (1999). Developing educational software components. *IEEE Computer*, 32(9):50–58.

[Sheth et al., 2006] Sheth, A., Verma, K., and Gomadam, K. (2006). Semantics to energize the full services spectrum. *Communications of the ACM*, 49(7):55–61.

[Shneiderman, 1997] Shneiderman, B. (1997). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Reading, MA, USA.

[Staab et al., 2001] Staab, S., Studer, R., Schnurr, H. P., and Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34.

[Stollberg et al., 2007] Stollberg, M., Keller, U., Lausen, H., and Heymans, S. (2007). Two-phase web service discovery based on rich functional descriptions. In *Proceedings of the Fourth European Semantic Web Conference (ESWC 2007)*, Innsbruck, Austria.

[van der Veer and van Welie, 2000] van der Veer, G. and van Welie, M. (2000). Task-based groupware design: Putting theory into practice. In Boyarski, D. and Kellog, A. W., editors, *Proceedings of the ACM SIGCHI Conference on Designing Interactive Systems: Processes, Practices, Methods and Techniques (DIS-00)*, pages 326–337, Nueva York, AZ, USA. ACM Press.

[Vega-Gorgojo et al., 2006] Vega-Gorgojo, G., Bote-Lorenzo, M. L., Gómez-Sánchez, E., Dimitriadis, Y. A., and Asensio-Pérez, J. I. (2006). A semantic approach to discovering learning services in grid-based collaborative systems. *Future Generation Computer Systems (FGCS)*, 22(6):709–719.

[Xu et al., 2003] Xu, Z., Yin, Z., and Saddik, A. E. (2003). A web services oriented framework for dynamic e-learning systems. In *Proceedings of the 2003 Canadian Conference on Electrical and Computer Engineering (CCECE 03)*, pages 943– 946, Montreal, Canada.

[Zhang et al., 2005] Zhang, L., Yu, Y., Zhou, J., Lin, C., and Yang, Y. (2005). An enhanced model for searching in semantic portals. In *Proceedings of the Fourteenth international conference on World Wide Web (WWW2005)*, pages 453–462, Chiba, Japan. ACM Press.