Annotating with uses: a promising way to the Semantic Web

Pierre-Antoine Champin, Yannick Prié, Alain Mille
(champin,yprie,amille)@lisi.univ-lyon1.fr
Cognition et Experience Team - LISI
University Claude Bernard of Lyon
http://experience.univ-lyon1.fr/

ABSTRACT

In Tim Berners-Lee's vision of the Semantic Web, software agents must be able to retrieve knowledge relevant to the end-user's task, despite the heterogeneity and scale issues which are inherent to the web. We argue in this paper that users' tasks are themselves quite numerous and various, so that they can not be all predicted, and hence implemented. Therefore, considering actual uses may be an efficient way of fulfilling users' unpredicted needs.

We propose a general approach for taking uses into account. Then we present how this approach was applied in two projects involving the authors: ARDECO, aiming at assisting the reuse of CAD documents, and RECIS, aiming at content-based retrieval of audiovisual documents. After a comparison of this approach with other works related to the Semantic Web, we conclude that any use of a resource should annotate this resource, and that such annotations, as they provide additional knowledge about the resource, should become part of the Semantic Web as resources of their own.

1. INTRODUCTION

In about a decade of years, the World Wide Web has become so popular as a mean of sharing information that the amount of available data is now overwhelming for users. Unfortunately, most of these data have been designed for humans rather than computers, and the help the latter can provide is limited. Tim Berners-Lee's vision of the Semantic Web [5] aims at improving the web by addressing this issue.

The Semantic Web must provide interoperability between heterogeneous agents. This means that not only data must be made available (as it is mostly the case today), but also metadata allowing a software agent to "learn" how to interpret the data: Document Type Definition, XML-Schema, RDF annotations... According to [2], the Semantic Web should even be able to provide justified answers to natural language questions, whereas current search engines provide lists of resources, supposed to contain the answer, in response to queries, supposed to relevantly represent the question though usually not expressed in any natural language. More generally, technological issues are not essential on the Semantic Web, human end-users are. This implies that *knowledge* rather than plain data must be retrieved, *i.e.* data which is relevant to the user's task. This also implies that social factors must be taken into account, such as privacy and trust, leading to the so called Web of Trust.

In this paper we will be more interested in the knowledge retrieval part. We believe that web resources can be used in many different ways by people along different tasks, and all these uses can not be listed *a priori*, if ever. On the other hand, eliciting the knowledge contained in a resource is always done with regard to a given task. Therefore, an adaptive agent should not *only* rely on the elicited knowledge, but also on the actual uses of the resource, so as to be able to provide informations relevant to an unpredicted task. In this paper, we present general principles enabling us to take uses into account.

In section 2, we present the notions underlying our approach. Then we present how this approach were applied in two projects: ARDECO (section 3), in the domain of mechanical design, involving structured documents from a CAD application, and RECIS (section 4), in the domain of multimedia retrieval, involving poorly structured audiovisual documents. Finally we discuss the advantages of the approach in section 5 and compare it to related works in section 6.

2. USES AND KNOWLEDGE BOXES

In this section, we present the notions underlying our approach, based on and extending the concepts presented in [18].

> *These notions will be demonstrated with a small example: a simple keywords based search engine.*
>
> *Please note first that we describe the uses of the search engine itself, rather than the uses of the browser, which is the interface between the user and the search engine. The browser is much more versatile and its use model too complex to fit our example.*

Knowledge Markup & Semantic Annotation Victoria B.C., Canada

2.1 Use model, task model

Any software system, including a web agent or a future Semantic Web agent, interacts directly or indirectly with a human agent: the "user". The latter can handle conceptually a number of "objects" thanks to the system. We call this the use model of the system.

Use model: The use model of a software system is the set of *objects* of the system the user can handle, and all the *operations* she can perform with these objects.

Any task involving the system can obviously be represented by means of elements of the use model. Although such models were once not explicitly available (they were embedded in the software code), this fact is now changing, thanks to the increasing use of object oriented design, and intermediate languages allowing "external" descriptions of objects, available to the user (especially for graphical user interfaces —cf. for example GladeXML[1]).

> *The use model of our example search engine is quite simple: it involves queries as sets of keywords and return a result list, where each result is a link to a web resource. Queries are submitted, results in the result list are traversed to the corresponding web page, and can be traversed back to the result list (usually by pressing the "back" button of the browser).*

Although the notion of *task model* has already been largely studied [8, 13], we here consider it with respect to the notion of use model : while a given task is performed, there are additional relations and constraints between objects of the use model. Hence we can define a task model as follows.

Task model: Every task model is a restriction of the use model: it specifies which properties of the objects are always verified during the corresponding task[2].

We wish to draw the reader's attention on the fact that the task models as we defined them are reporting tasks rather than prescribing them. Hence our definition is more general than the ones usually proposed, while including them: our task models could possibly be prescriptive, though they would be less explicative (cf. 2.2).

> *We describe here two typical tasks in using a search engine, by means of the use model described above. The task of scanning the results consists, given a result list, in traversing the links of the result list forward (to the resource) and backward (to the result list). The last link may be traversed forward only (in which case the result is considered a success) or forward and backward (in which case the result may be considered a failure).*
>
> *Another typical task is the one of refining a query: given a result list returned by submitting query Q_1, a new query Q_2 is submitted, containing Q_1 plus some other keywords. The goal of such a task is to reduce the number of results in the result list.*

2.2 Use cases

The set of objects "involved" in the user's task has its *state* changed at each operation performed by the user. Therefore we can represent a trace of any use of the software as a sequence of states and operations. These traces alone are hard to exploit (usually, data mining techniques are used with that kind of material, in order to extract pieces of knowledge), hence the interest of annotating parts of it with *use cases*[3].

Use case: A use case is a subsequence of the states/operations trace which is an instance of a task model. We say that the task model explains the use case.

Here we see that prescriptive task models, as those mentioned above, are not very likely to explain many use cases, since few users follow them exactly. On the other hand, more general task models will be matched more often by use traces.

> *Figure 1 represents a trace of the use of our search engine. The sequence 1–2 is an instance of the task model query refining, hence it corresponds to this task. Sequences 2–5 and 6–9 correspond to result scanning tasks, resulting in a failure for the former (because it ends with a backward traversal), and in a success for the latter (ending with a forward traversal).*
>
> *Note that the sequence 5–6 is not explained by any task model: it is not a query refining as we defined it, since the new query does not contain all the keywords from the previous one[4]. This is therefore an example of unpredicted task, which can not be explained by available task models.*

2.3 Knowledge box

We will not define the term *knowledge box* (KBx[5]) as we did with the other notions above, because this term covers

[1] ⟨URL:http://glade.gnome.org/⟩, XML language for describing graphical interfaces, interpreted at runtime.

[2] This is of course a system-centered view. From the point of view of the user, tasks are related to goals, which are often implicit and external to the use model. It is also noteworthy that the use model of a system can itself be considered as a task model of a larger system —for example, the use model of a word processing application is a task model in a productivity suite.

[3] These "use cases" must not be confused with UML use cases [20], which are generic and then more akin to our task models.

[4] From the point of view of the user, on the other hand, this is a query refining indeed: the reader might have guessed that our user was looking for pictures *painted by* Picasso, while the second query might have returned photographs *of* Picasso.

[5] The acronym KB being often used for "Knowledge *Base*".

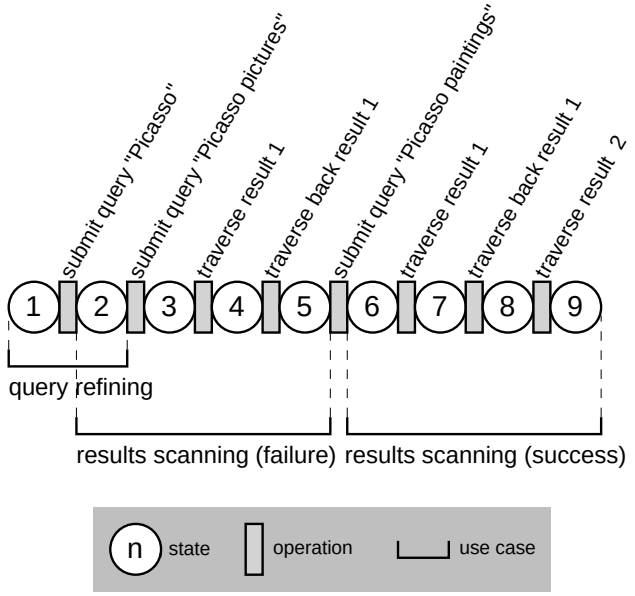

Figure 1: Example of a trace with use cases instantiating task models.

the same things as the term "resource". As a matter of fact, any KBx available on the Semantic Web is obviously a resource (everything on the web is), and every resource can be a source of knowledge to somebody. However, the name "knowledge box" allows us to insist on some interesting features.

First, a resource may have several forms, depending on many factors: the moment in time (versions, corrections), the medium (screen, audio), the encoding format (html, pdf), etc. On the other hand, some resources have no retrievable form at all but can be used in other ways (like mailto: resources). Each form, each use, is suitable to some users' tasks, and can provide them with some knowledge, hence the term "knowledge" box.

Second, the word "box" carries the meaning of a *container* of knowledge. But it also implies that the KBx could be a "white-box" or a "black-box". In the former case, the structure of the KBx is known, and some knowledge can be extracted or inferred by a software agent; but in the latter case, the KBx has no known structure and is considered opaque to the software agent. Only the human end-user has the ability to extract the relevant knowledge. As a matter of fact, and as we said in introduction, all the uses of a KBx can not be predicted from the start, nor can the whole knowledge one could extract from the KBx. Knowledge white-boxes can then become black-boxes in the context of unpredicted tasks.

This does not mean however that our approach disregard any ability to automatically extract knowledge from KBx's. Indeed, our first example, related to mechanical design, relies to a large extent on the internal structure of KBx's.

> *The KBx's primarily involved in our search engine scenario are of course web resources. These KBx's are not strictly opaque to the search engine, since it can extract keywords from them, but on the other hand, keywords are usually not the only knowledge users are looking for. We can see here that there is no strict dichotomy between knowledge black-boxes and knowledge white-boxes.*

The main idea of our approach is that use cases provide the agent with concrete examples where KBx were used. In similar contexts, *i.e.* similar use cases, it is likely that the same KBx will prove relevant for the user (even if this relevance is not directly "understood" by the software agent). This can be performed using the Case Based Reasoning (CBR) paradigm, whose mechanisms have been largely studied in the artificial intelligence community [1, 15]. We present in the following sections two applications of the above approach in different domains.

3. EXAMPLE 1: ARDECO

Designers of complex systems are inclined to reuse previous design artifacts or previous design processes, since adapting an old design is often easier and quicker than designing from scratch; thus it enables them to reduce the duration and cost of the design activity. On the other hand, retrieving reusable material is not always easy, all the more so as designing for reuse is often perceived by designers as over-constraining.

Although most design activities are now computer aided, CAD systems do not provide users with efficient assistance to reuse. The aim of the ARDECO[6] project is to set up such an assistance system: a software agent enabling designers to retrieve, from a collection of CAD documents on the company intranet, reusable material for their current task. This document retrieval function makes this kind of agent similar to Semantic Web agents.

3.1 Use model and task models in ARDECO

3.1.1 Use model and knowledge boxes

Since ARDECO focuses on a specific mechanical CAD application (namely Dassault Systemes' CATIA), the use model is precisely defined, and the knowledge boxes, namely design artifacts, are well structured: they are represented in CAD documents (files saved by the application) as a specification tree, where each element is an instance of a class, has a number of components, attributes, plus possibly additional references to other elements in the tree. The activity of the user consists in adding, modifying and removing elements in the specification tree, either directly or through the geometrical representation of the designed artifact (cf. figure 2).

3.1.2 Task model

Although the use model is precisely defined, task models are not. As a matter of fact, the number of classes in the use model is quite large, and so is the number of possible relations between elements. Furthermore the design activity does in essence involve the user's creativity. Therefore the number of possible tasks is considerable, and task models can not be exhaustively described —at least sufficiently specific task models, so as to be reusable (some tasks may

[6]ARDECO is a French acronym standing for "Assisted Reuse of Design Episodes". As a part of the PROSPER program of the French national center of scientific research, it involves researchers in computer sciences and cognitive psychology, as well as the industrial partner Dassault Systemes.

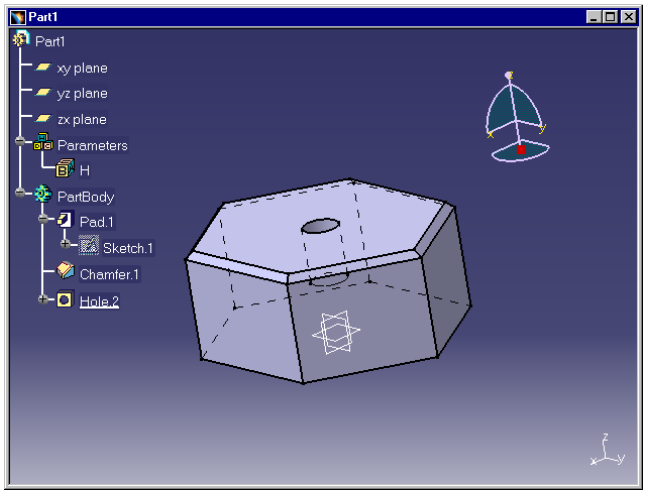

Figure 2: Catia screenshot, showing both the hierarchical structure and geometrical representation of the designed artifact.

be identified in the application, but they are far too generic, *e.g.* 2D sketching).

3.2 Use cases: design episodes

Since task models are not available to explain use cases, the most extensive use of available knowledge has to be made: by exploiting the richness of the use model (hierarchical structure of design artifacts), but also by interacting with the user (not so as to disturb her, however). We decided to use the notion of *design episode* in order to do so; this notion is based on works in psychology and ergonomics, and is defined in our domain as "a part of the design activity between the moment a goal is identified and the moment that goal is considered reached". This definition allows us to consider episodes as use cases, instance of a task model (*a priori* unknown to the system).

Instrumenting CATIA allows designers to annotate parts of their activity, either by declaring explicitly that an episode bound is reached, or by indicating what goal they are trying to achieve at a given moment. Finally, observations of end-users working with CATIA, performed by cognitive psychology researchers [6], have provided us with behavioral cues, enabling the system to automatically detect some episode bounds and annotate the design activity consequently. Hence annotations are performed on the fly, and actually from the user's viewpoint.

3.3 Knowledge boxes retrieval

As we said, we can consider design episodes as instances of task models. The corresponding tasks are not known, or we have to consider that every episode is the only instance of a very specific task model. However, thanks to the rich use model and to the transparency of KBx's, episodes, and hence their underlying task models, can be compared with each other on the basis of static features (comparing states) and dynamic features (comparing the changes performed in each episode).

The trace of the activity of the user is continuously stored; when requested by the user, the system compares the current episode to others in an episode base. It retrieves the most similar elements: states or episodes, depending on the features (static or dynamic) most contributing to the similarity.

Furthermore, elements actually reused by the user are linked to the current episode, thus defining a "reuse case". These links between reusable material and reuse cases annotate episodes themselves, and can be use to improve the relevance of the retrieval (episodes considered similar once have chances to be considered so later) and to suggest adaptation [7].

Hence each structure level provides a kind of KBx: states as a structure of CAD specifications, episodes as annotations of states, reuse cases as annotations of episodes. An under-development prototype represents all those structures in a single graph thanks to the RDF language [16], which allows the merger of different level vocabularies.

4. EXAMPLE 2: RECIS

Our second example takes place in the field of audiovisual information retrieval, with the RECIS project[7]. This project aims at building new tools for better access to and use of audiovisual material with a mixed approach: facilitate indexing with multimedia information extraction, and exploit search sessions so as to help users to quickly find successful queries. This makes it more directly related to the Semantic Web than the previous example.

4.1 Audiovisual documents as knowledge black-boxes

Audiovisual (AV) documents mainly differ from textual documents to the extend that they do not have, until now, any explicit documentary structure. The only acknowledged structure one could think of is basically a moving images stream along with audio streams.

Of course, that characteristics may be transient, and works such as the MPEG-4 or MPEG-7 initiatives aim at setting up documentary standards for structured description of AV documents. But the relative absence of structure has also a reason: innovating usages of AV documents (navigating, indexing, search, etc.) still remain to be invented. Hence no "killer application" did allow the emergence and stabilization of sufficiently shared ways of describing yet: until now, audiovisual documents remain knowledge black-boxes.

One consequence of that fact is that, whatever usage of AV document one wishes to set up (of course differing from simple visualization), she has to make a document structure explicit in order to allow this usage. In the general case, she has to do that from scratch[8].

Hence, so as to allow automated agents to gain control over audiovisual documents, and enrich access and query tools, it appears to us necessary to set up models that can adapt to various uses. These models should allow us to

[7]RECIS, standing for "Content Exploration and Retrieval with Image and Sound", is a French project, part of the RNRT (French telecommunication research network) program. It involves LISI, INRIA and France Télécom.

[8]Another consequence — particularly interesting for the document community: we can specify that any structure set up in that scope is *semantic*, and that there is no *a priori* "canonical" documentary structure for AV documents. This contrasts with textual documents, whose typographic tradition has last for centuries.

consider an unlimited number of structures for a document or a set of documents, which should take the general form of a graph.

4.2 Use model and task models in RECIS

The description model we introduce in this section was presented in [19] and extended in [4], and is named AI-Strata. We will only present here what corresponds to our general approach.

4.2.1 Use model

An *annotation element* (AE) is basically a term[9], like ⟨*Clinton*⟩ or ⟨*sax*⟩, annotating a fragment of an audiovisual document named *audiovisual unit* (AVU). The main way of enhancing the structure of AE's is to put them in relation with each other[10]. Any two AE's can be put in relation (which is called *elementary relation*), even, in particular, if they annotate different AVU's; hence the name of the model: Annotation Interconnected Strata (AI-Strata). It becomes possible to set up an annotation graph, with as many annotated AVU's as there are points of view (and hence uses) of the AV document.

There can be many AE's which are occurrences of the same term. They are specified thanks to *abstract annotation elements* (AAE). One AAE uniquely defines the term of the related AE's. Annotation elements, abstract annotation elements and audiovisual units define the use model of AI-Strata, whose elements are structured as a direct labeled graph, into which information will be retrieved.

An element of the graph is said to be in the context of another one if there exists a path between them in the graph. Searching a node is performed by describing its desired context as a graph pattern, which we call a *potential graph* – and is similar to query graphs in the object oriented domain. Any search can hence be computed by partial subgraph isomorphism. Our point of view is that any exploitation of the graph (*i.e.* element search in it) should be considered in a contextual way, since the meaning of an annotation term is determined by its context (⟨*Clinton*⟩ in relation with ⟨*sax*⟩ is not interpreted the same way as ⟨*Clinton*⟩ alone).

4.2.2 Task models

Abstract annotation elements are grouped into *analysis dimensions* (AD) defining classes of terms. These classes are not necessarily disjoint (*e.g.* ⟨*AD:Politicians*⟩ or ⟨*AD:TVShows*⟩ may have ⟨*Clinton*⟩ in common). They group AAE that should be used in the same way for some descriptions or, in other words, for some task. Hence they represent (very primitive) task models.

Annotation can be directed by recommending the use of some particular AD, but also by *description schemes* (DS), which also specify the relations to be set up between annotation elements. For instance, a description scheme ⟨*DS:TVNews*⟩ would specify that an AVU be annotated with an AE from analysis dimension ⟨*AD:Politicians*⟩, put in relation with another AE, from ⟨*AD:actions*⟩, and annotating an AVU from the same stream (like in figure 3, AE's ⟨*Clinton*⟩ and ⟨*Hand Shaking*⟩). Description schemes are more elaborate task models than analysis dimensions.

4.3 Knowledge boxes retrieval

Since analysis dimensions and description schemes represent the way users are the most likely to annotate, they are also to be used to retrieve annotated streams. However, AD and DS can be more or less shared between communities of users. For instance institutions like broadcast stations have more standardized ways of describing than individual users. The AI-Strata model allows the description of any piece of AV documents in many ways: from describing documents as wholes to splitting them into structured AVU's, and from simple keyword descriptions to "classical" knowledge representations (if analysis dimensions and description schemes constitute a formal ontology, with terms having a strictly defined semantics). Any setting up of an annotation graph conforming to a task model (AD or DS) is a use case of RECIS [4].

An important point is that however strict the task model is, users can always set up additional relations between AE's, which are not specified in a given description scheme, or they can merge several description schemes having been developed independently. Hence it becomes possible to put in the annotation graph, as well as in potential graphs for retrieval, more knowledge that the one captured in the task models —this practice is similar to the one described with ARDECO, in the previous section.

Although the initial goal of AI-Strata was to retrieve AV documents thanks to the annotation graph, potential graphs allow users to retrieve other nodes as well, such as AE's or AAE's, depending on their task, since the context of each node may carry as much information as the annotated AV-documents. Hence AV documents are no more the only KBx's in the model: any partial subgraph of the annotation graph may be considered as a KBx.

A prototype has been developed in the context of RECIS. It allows to visualize and annotate AV streams, as well as searching a distributed annotation graph. This prototype represents annotations and queries with XML and XLink [11]. Current efforts are oriented at developing a web-based interface to the system. We are also working on an RDF representation of the AI-Strata model.

5. DISCUSSION

The usage based approach we propose has been applied in the two projects presented above; it is also being applied in other domains[11], like distant learning (PIXED project) and knowledge management in collaborative design (OSCAR project). The common feature of these projects is the wide range of possibilities let to the user, hence the necessity of our approach, in order for the system to answer relevantly to new uses. Table 1 is a summary of the way the approach has been applied in ARDECO and RECIS.

Both projects are quite different with regard to the amount of available knowledge at each level. Indeed ARDECO provides a lot of knowledge in the use model, since it is bound to a specific application (CATIA), while on the contrary the use model of RECIS is minimal. On the other hand, RECIS allows explicit descriptions of task models by means

[9] This model is based on linguistics rather than object-oriented modeling, so AE's do not have an intrinsic semantic as object identifier; they *just* stand as terms.

[10] Another less important way is to give attributes to an AE, like for instance ⟨*Shot,Keyframe=shot127.jpg*⟩, but this could also be achieved with elementary relations.

[11] ⟨URL:http://experience.univ-lyon1.fr/projets.html⟩

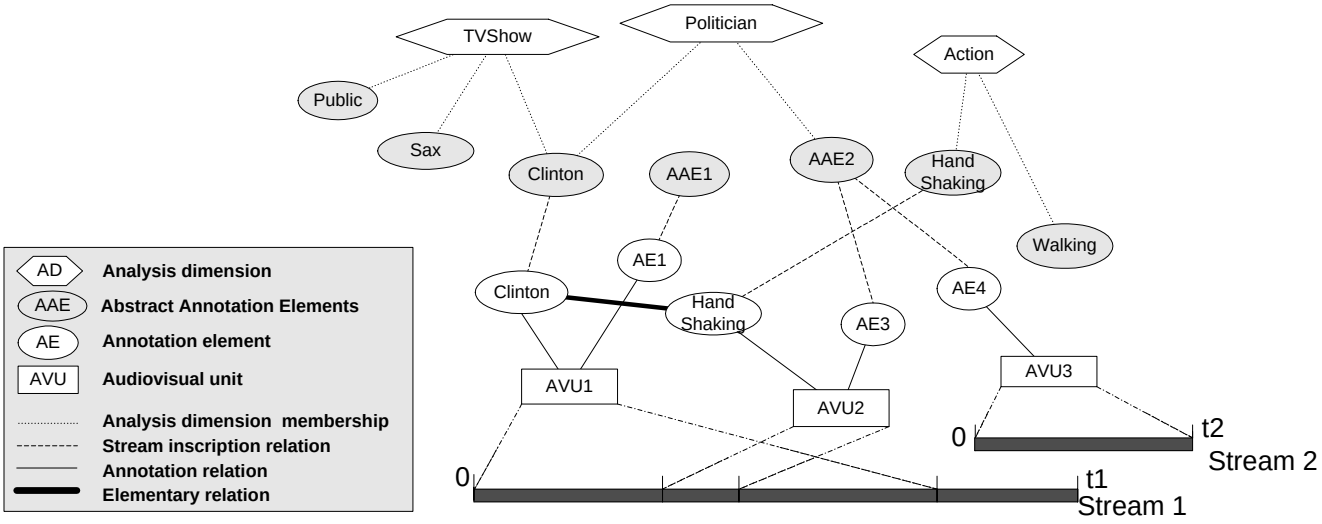

Figure 3: An example of annotated audiovisual stream. The AV stream is cut into audiovisual units that are annotated with annotation elements coming out of analysis dimensions.

	ARDECO	RECIS
Use model	CATIA use model	AV units, Annotation elements, Abstract annotation elements
Task models	Unknown, instantiated as episodes	Analysis dimensions, Description schemes
Use cases	Episodes	Annotation graph
KBx's	States, episodes, reuse cases	AV documents, any partial subgraph of the annotation graph

Table 1: ARDECO and RECIS application of the usage based framework

of analysis dimensions and description schemes, while tasks can only be delimited in ARDECO, but not elicited. This resulted in quite different implementations, with a different management of the available resources, as knowledge white-boxes or black-boxes.

It is worth noting, however, that the software agents built with our approach do not primarily rely on the "color" of the box, but rather on the external description of its uses. The base principle is that a KBx may be useful in a situation if it has already been used in a similar situation, *i.e.* if it has been annotated in the same way. The relevance of the result is no longer bound to the capacity of the agent to "understand" the resource *nor* the task of the user, but only on its capacity to compare use based annotations.

A second interest is that use cases provide information to users themselves (provided that they are presented in an appropriate form): the annotations set up to describe uses become KBx's of their own, as important as the initial resources. That is why in the presented applications, knowledge retrieval is not limited to CAD or AV documents, but allows the search of any element of the use model.

6. RELATED WORKS

Most current works related to the Semantic Web aim at providing the interoperability it requires, while coping with problems inherent to the web: it is massive, open (in contrast with the "closed world" hypothesis), and dynamic (constantly changing) [14]. It is commonly agreed that a way of achieving this is recourse to ontologies for annotating resources; ontologies are formal vocabularies with well defined relations between their terms, shared by a community of users like the *Ontogroups* of the ONTOBROKER system [10]; this makes them inevitably oriented toward a specific task or task family.

In the SHOE system [14], ontologies can be defined and extended in any HTML document, and each statement is associated with its *claimant* (the entity responsible for the statement), in order to be able to manage contradictions or trust issues.

Moreover, ontologies can provide agents with powerful inference capacities related to the corresponding task. For example, the language OIL [12] is particularly focusing on the complexity of inference algorithms, thus addressing the issue of the web being massive. It is based on description logics, whose inference algorithms have been largely studied. OIL has recently been merged with the DAML [9] project, aiming at describing ontologies in RDF.

However some works are relying on uses, especially a minimal use model of the web involving resources (being created) and links (being set up or traversed). As a matter of fact, the web is primarily a set of resources, interconnected by hyperlinks. This is how it grew up with HTML, and this is what the Semantic Web is intended to emerge from.

Previous search engines only relied on the content of HTML documents to evaluate their relevance to a given keyword based query (like in the running example of section 2); their noise rate was huge. On the contrary, the search engine GOOGLE[12] relies on the text of the hyperlinks to these pages. The rationale beneath this approach is that a page is relevant with respect to a keyword when other people point to this page with this keyword, and it has proved quite efficient. We argue that this successful approach actually relies on a kind of uses of the pages (setting up a link to them) rather than their content. A step further would be to take into account more structured links like the ones proposed in RDF [16], XLink [11] or the Annotea project [3].

[17] give a good overview of works on *implicit feedback* in the domain of information searching: this consists in observing the user's behavior and indexing document by their uses rather than their content. They give a classification of general behaviors in the activity of searching information, which can be considered, in our model, as very general task models.

We believe that uses based approaches complement ontology based approaches: while the latter provide resource descriptions oriented *a priori* toward a given task, along with powerful inference capabilities, ours can not claim the same level of logical inferences, but is adaptable *a posteriori* to any unpredicted usage.

7. CONCLUSION

In this paper, we propose general principles aiming at taking uses into account in the design of software agents for the Semantic Web. We consider resources as knowledge boxes, which allows us to focus on the fact that they may contain some knowledge available only to the human end-user, not to the software agent. Hence the latter must rather rely on the use model, possibly annotated by task models explaining it. It allows the agent to handle previously unknown tasks. Moreover, use cases structures may also provide the user with additional knowledge, and hence become knowledge boxes of their own, because meta-knowledge is knowledge too. Two applications of this approach are presented, in the domains of mechanical design and audiovisual documents retrieval.

These examples, as well as the following discussion, show how the proposed approach may be applied in various domains. As it was told mentioned in section 5, we are currently applying it to other projects, and we are working on a methodological framework for its deployment, with implementation guidelines as well as XML and/or RDF schemas to represent use models, task models and use cases.

Another subject of interest related to our approach is to work on the elicitation of task models emerging from recurrent use cases. As a matter of fact, those task models would represent the effective uses of the system, and could even help to build *a posteriori* ontologies for these discovered tasks.

8. REFERENCES

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AICom*, 7(1):39–59, 1994. ⟨URL:http://www.iiia.csic.es/People/enric/AICom.html⟩.

[2] H. Akkermans. Future of KM Technology in the Knowledge Economy. Invited talk at WM'2001, Baden-Baden, March 2001. ⟨URL:http://wm2001.aifb.uni-karlsruhe.de/InvitedTalks/⟩.

[3] The Annotea project. ⟨URL:http://www.w3.org/2001/Annotea/⟩.

[12] ⟨URL:http://www.google.com⟩

[4] A. Benel, E. Egyed-Zsigmond, Y. Prié, S. Calabretto, A. Mille, and J.-M. Pinon. Truth in the Digital Library: From Ontological to Hermeneutical Systems. In *5th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, volume 2163 of *LNCS*, pages 366–377, Darmstadt, Germany, September 2001. Springer-Verlag.

[5] T. Berners-Lee and M. Fischetti. *Weaving the Web.* Harper San Fransisco, 1999.

[6] P. Bougé, F. Détienne, and L. D. Cesare. Épisodes de conception : une étude ergonomique pour le recueil de "cas". In *Ingénierie des Connaissances (Knowledge Engineering) 2001*, Grenoble, June 2001. *in French.*

[7] P.-A. Champin. A model to represent design episodes for reuse assistance with interactive case-based reasoning. In I. Vollrath, S. Schmitt, and U. Reimer, editors, *GWCBR'2001*, pages 189–197, Baden-Baden, Germany, March 2001.

[8] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins. Ontology of tasks and methods. *IEEE Intelligent Systems*, May 1999. ⟨URL:http://www.cis.ohio-state.edu/~chandra/Ontology-of-Tasks-Methods.PDF⟩.

[9] The Darpa Agent Markup Language (DAML) project. ⟨URL:http://www.daml.org/⟩.

[10] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. M. et al., editor, *Semantic Issues in Multimedia Systems. Proceedings of DS-8*, pages 351–369. Kluwer Academic Publisher, Boston, 1999.

[11] S. DeRose, E. Maler, D. Orchard, and B. Trafford. XML Linking Language (XLink) Version 1.0. W3C candidaTE recommendation, July 2000. ⟨URL:http://www.w3.org/TR/2000/CR-xlink-20000703⟩.

[12] D. Fensel, I. Horrocks, F. V. Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a Nutshell. In R. D. et al., editor, *Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW-2000).* Lecture Notes in Artificial Intelligence, LNAI, Springer-Verlag, October 2000.

[13] D. Fensel, E. Motta, V. R. Benjamins, S. Decker, M. Gaspari, R. Groenboom, W. Grosso, M. Musen, E. Plaza, G. Schreiber, R. Studer, and B. Wielinga. The Unified Problem-solving Method Development Language UPML. Delivrable, University of Karlsruhe, Institute AIFB, 1999.

[14] J. Heflin, J. Hendler, and S. Luke. SHOE: A Knowledge Representation Language for Internet Applications. Technical report, Dept. of Computer Science, University of Maryland at College Park, 1999. ⟨URL:http://www.cs.umd.edu/projects/plus/SHOE/pubs/techrpt99.pdf⟩.

[15] J. Kolodner. *Case Based Reasoning.* Morgan Kaufmann Publishers, 1993.

[16] O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C recommendation, February 1999. ⟨URL:http://www.w3.org/TR/1999/REC-rdf-syntax-19990222⟩.

[17] D. W. Oard and J. Kim. Modeling Information COntent Using Observable Behavior. In *American Society for Information Science and Technology*, Washinton DC, November 2001.

[18] Y. Prié, A. Mille, and J.-M. Pinon. Modèle d'utilisation et modèles de tâches pour l'assistance à l'utilisateur basée sur l'expérience : le cas d'un système d'information audiovisuelle. In *Ingénierie des Connaissances (Knowledge Engineering) 1999*, pages 21–30, Palaiseau, June 1999. *in French.*

[19] Y. Prié, A. Mille, and J.-M. Pinon. A Context-Based Audiovisual Representation Model for Audiovisual Information Systems. In *International and Interdisciplinary Conference on Modeling and using Context*, volume 1688 of *LNAI, Springer-Verlag.*, pages 296–309, Trento, September 1999.

[20] Unified modeling language (uml). ⟨URL:http://www.omg.org/technology/uml/⟩.