Semantic Layer Architecture for an Educational Expert System in Computer History

Serge Linckels, Christoph Meinel

Abstract – **In this paper we present the results of our research about a semantic layer architecture for an educational tool. It uses technologies from the Semantic Web, and is based on the original Semantic Web layer architecture, but respects recent critiques and proposals of that architecture. Our solution is composed of four layers with respect to a distributed system like the Internet. Each layer can be located on a different computer and on different platforms. Furthermore, the details about the representation and encoding of the heterogeneous knowledge, as well as the reasoning mechanism and the communication between the different layers, are transparent for the user. In order to test whether our solution is useful in schools, we implemented it in an educational tool, CHESt (*Computer History Expert System*) that offers a semantic search engine to the user. The facility of interacting with the tool, by means of natural language, and the multimedia aspect of the answers returned by the system, make CHESt a useful e-learning tool in everyday classes.**

Index Terms – **Information retrieval, knowledge representation, natural language interfaces, semantic networks.**

I. Introduction

The WWW is today accepted in schools as modern didactical tool with its advantages and disadvantages. If the tremendous amount of information on the web is on the one hand a potentially interesting source of knowledge, it is on the other hand a dangerous pitfall, mainly due to the weakness of the search engines. Most of them are not able to deliver only pertinent and secure results; not pertinent means that the search engine does not understand the meaning of the user's question; not secure means that the search engine is not able to guarantee that the delivered result is true and correct. Today, using a search engine on the web is often the exercise of filtering *noise* from the resulting list of links; it is the famous story about the search of the needle in the haystack. A good e-learning tool must be able to return secure and pertinent information to the user, without assuming that (s)he is expert in expressing her/his question in a computer optimized way, for example by using Boolean operators. Our aim is to create an "intelligent" tool, that understands the students' questions, and that returns only pertinent documents from a knowledge base. Our prototype CHESt (*Computer History Expert System*) can be used as a complement to classical education. For example, the teacher can use it in class to introduce a new topic or to promote group work, students can use it to do their homework and it offers excellent possibilities for distant learning. A first draft of CHESt, with a common keyword search engine, was presented in [5]. Reflections about a possible semantic search engine were firstly presented in [6]. Pedagogical analyses of the use of such an "intelligent" e-learning tool in every-day classes were published in [7]. In more general terms, we try to create the formal and technical base for a polyvalent and pragmatic educational tool. For this, the following requirements must be fulfilled:

- The human-machine interface is simple and easy. Hence, the complexity of its inference engine and retrieval algorithms is transparent to the user.
- The semantic search engine is platform independent.
- The tool is operational without special configuration and installation procedures on the user's computer.
- Distant and local access facilities are provided.

Finding a solution to all these requirements was the aim of our recent research efforts, which are based on experiences and technologies from the field of Computational Linguistics and the Semantic Web. In this paper we focus on the aspect of the layer architecture that is implemented in the latest version of CHESt.

We discuss briefly the layer architecture of the Semantic Web in section II. In section III, we present our simplified layer architecture that is implemented in CHESt. Sections IV to VII present the four layers of our architecture: Knowledge, Inference, Communication, and Presentation Layers. A case study of our tool will be presented in section VIII. Section IX presents some interesting and related projects. We conclude with a discussion of the advantages and weaknesses of our layer architecture in section X.

II. Discussing the Semantic Web's Layer Architecture

The vision of the Semantic Web put together so many experts from so many different existing and new domains (information retrieval, artificial intelligence, knowledge management, etc.) as rarely, if ever, before in the history of computer science. Although our project is not directly related to the Semantic Web, it builds on technologies that stem from this field. Therefore, we start with a small introduction to the basic properties of the Semantic Web taken from [1]:

Submitted for the 2004 International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA), Luxembourg, in cooperation with the IEEE Computer Society. Manuscript received September 15, 2004.

S. Linckels is with University of Trier, Germany, Department for Theoretical Computer Science and New Applications, http://www.linckels.lu, e-mail: linckels@TI.uni-trier.de.

Ch. Meinel is with Hasso-Plattner Institute, University of Pottsdam, Germany, http://www.hpi.uni-potsdam.de/, e-mail: Meinel@hpi.uni-potsdam.de.

- A common language to structure data and rules. This should allow to create ontologies which allow humans and machines to have the same understanding of shared concepts.
- A distributed system like the Internet, thus completely platform independent. There is and will be no central control of the data or the software, because such a system will rapidly become unmanageable.
- A layer architecture, the so called *Layer Cake* [4] (see figure 1). On a basic level, the knowledge is semantically annotated and ontology languages build on this representation, which allow programs (for example: agents or search engines) to reason about the knowledge.

Complex computer systems, for example operating systems or expert systems are built on a rigorous and well-reasoned layer architecture. The Semantic Web is based on such an architecture which, however seems to be quite complex and idealistic. The discussion of all details, critics and proposals about that subject are outside the scope of this paper. We will only summarize the two most significant ones for our project.

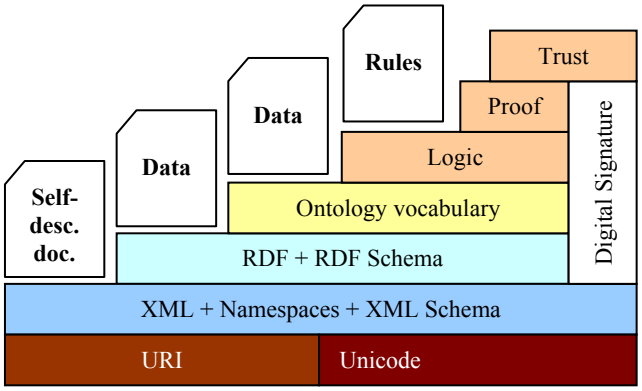

Fig. 1. The semantic layer architecture. The top three layers exist only in theory. The Ontology Layer is heavily criticized. The RDF and XML layers are used today.

Firstly, one major criticism is its complexity which makes the implementation unrealistic [8, 9]. For example, the idea of building ontology languages like OWL [12] on top of RDF Schema [11] raises the complexity of the system that implements this architecture. Secondly, the syntactic and semantic characterization underlying RDF(S) differs significantly from the syntax and semantics of most first-order logical languages [9]. A one-to-one data exchange between the RDF- and Ontology Layer is not free of ambiguities. For example, OWL's theory of classes clashes with the underlying principles of RDF(S) in the same syntax and extended semantics layer [8, 9].

A lot of interesting simplifications and solutions to the Semantic Web layer architecture are proposed in literature. Most of them propose to simplify or to suppress layers. In [3] for example, a solution is proposed to merge the Ontology- and RDF Layers to one, and to extend it with the more powerful RDFS(FA) formalism [10].

III. Our Semantic Layer Approach

Inspired by the Semantic Web technologies and theories, we developed a semantic retrieval system around a straight-forward layer architecture. In fact, there are mainly two ways to improve the layering of the Semantic Web [3]:

- To change some aspects in the design principles of higher ontology languages built on RDF(S) and to leave the RDF(S) specifications unaltered.
- To modify the specifications of RDF(S) in order to make them more compatible with the underlying reasoning formalism of ontology languages like OWL(DL).

Our layer architecture defends clearly the first proposal and keeps the RDF(S) specifications untouched. We built on top of our RDF(S) knowledge layer a native inference engine with adapted and optimized reasoning mechanisms for our domain. In general, our layer architecture is composed of four main layers (see figure 2), each being able to hold sub-layers and modules.

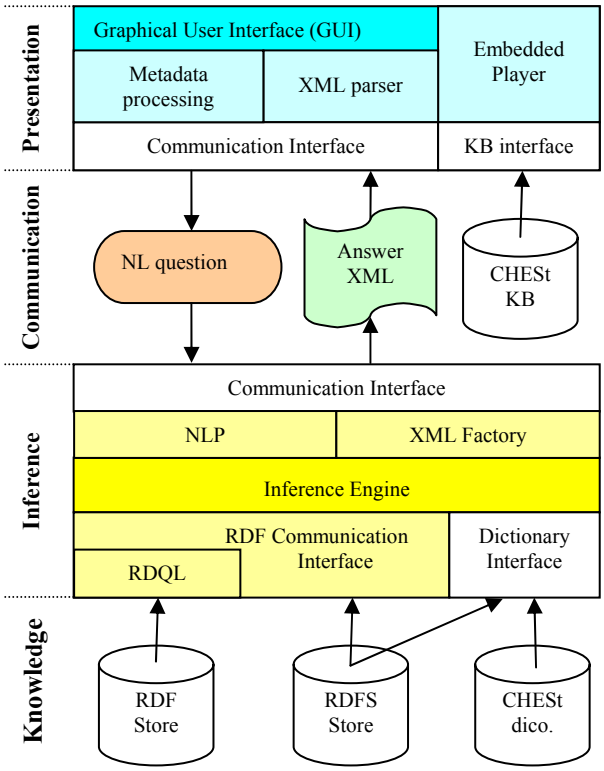

Fig. 2. Schema of our semantic layer architecture with its four main layers: Knowledge, Inference, Communication, and Presentation. Each layer is composed of sub-layers and modules.

The **Knowledge Layer** regroups all knowledge resources that are used for the reasoning processes. It contains the RDF(S) stores for the semantic description of the multimedia clips and the domain language (CHESt dictionary).

The **Inference Layer** is technically the most important, because it implements the semantic retrieval system. It is composed of the natural language processing (NLP) module, the inference engine for reasoning over the knowledge base with respect to the user question, and the XML factory module for generating and encoding the system's answer.

The **Communication Layer** specifies the exchange of information between the distributed Inference- and Presentation Layers. It must ensure a transparent but error-free communication.

The **Presentation Layer** implements the Human-Machine Interface, and thus needs an ergonomically adapted look. It allows the user to freely formulate a question in natural language (NL). The result, a commented list of multimedia clips, is then presented to the user. Each clip can be viewed in an embedded player.

IV. KNOWLEDGE LAYER

The Knowledge Layer is the set of data sources which are accessed by the upper inference engine for reasoning about the knowledge. The raw data in the knowledge base can be of any form; in our case multimedia sequences. Indeed, interpretations over the knowledge are only promising if its content is semantically annotated with sufficient metadata [30]. Therefore, it seems evident to use a consistent semantic representation of the knowledge, and to offer a uniform interface to the upper layer.

A. Sub-Layer of Raw Data

The raw material for CHESt was hundreds of slides in electronic form about computer history. These slides were used to produce the multimedia sequences; we prefer the expression *clip*. We produced more than 300 clips with a special tool called tele-TASK [13, 14, 15]. It allows creating one well structured stream of data, in our case *RealMedia* files (`.rm`). Each clip presents multimedia information in three windows at the same time (see figure 6):

- A window showing a video sequence (with the corresponding audio).
- A window showing the desktop of the presenter's computer.
- A window which can be used to display further information, for example pictures, hyperlinks, book references, etc.

More details about the encoding of the multimedia sequences can be found in [7]. All clips are identified by a *Unique Resource Identifier* (URI), and can be located on a distant server (accessed by file sharing or streaming), or on the local machine of the user (for example on a CD-ROM). The different options and versions of CHESt are discussed in section VIII.

B. Semantic Annotation of the Clips

Classical retrieval systems work by indexing important words from the documents' corpuses [30]. In our case however, the clips are stored in a non-textual form. Therefore, the search for pertinent documents cannot be done by processing the documents' content. Furthermore, we are not interested in the documents' content but only in the semantics of the document (what it is about). Thus, all clips must be described with metadata which explain the meaning of its content.

We used the W3C recommendation RDF(S) [11] to describe every document in the knowledge base with metadata. RDF is a mechanism for recording statements about resources so that machines can easily interpret the statements [16]. In our case, a resource is a multimedia clip. A bit of knowledge about a resource is represented by a *triple*, which is three pieces of information: a subject (resource), a property (predicate) and a value (object). The resources were categorized in a taxonomy that is in our case about computer history. Details can be found in [6]. As we mostly used elements from two popular namespaces, *Dublin Core* [17] and *vCard* [18], we only needed to define few new elements with RDF Schema. We also defined several rules how the elements should be used. The complete CHESt RDF Schema definition can be found at [19].

There should be enough metadata to describe semantically each clip. Metadata are in fact nothing else than RDF statements (triples) about the clips (resources). With our concept to use short clips instead of long sequences, we have the great advantage that the meaning of one clip can be described with few metadata. We encoded the metadata in RDF/XML. An example is presented in figure 3.

```
<chest:EComponent rdf:about="…transistor.rm">
  <DC:title>Transistor</DC:title>
  <DC:date>1947</DC:date>
  <DC:creator rdf:resource="…shockley.rm" />
  <DC:creator rdf:resource="…bardeen.rm" />
  <DC:creator rdf:resource="…brattain.rm" />
</chest:EComponent>
```

Fig. 3. Example of a semantic annotation of a multimedia clip with RDF. It describes a resource (a multimedia clip) about the transistor, which is classified in the taxonomy as electronic component (EComponent). Metadata are the title (in a human readable form), the year the invention was made, and the list of the inventors. The example shows that a resource exists for each of the inventors.

The creation of the RDF description for a whole knowledge base is often a painful task if it is done manually. The automatic annotation and the information extraction are hot topics in computer science, especially in the field of web engineering [21]. This task is even harder when dealing with multimedia sequences [20]. In our implementation, we used templates to create the documents so that metadata could be found and extracted more easily. More than 85% of the RDF description was generated automatically. Details about that solution can be found in [6].

V. INFERENCE LAYER

The inference layer is the most important in our expert system, because all the reasoning is done at this stage. In a simplified view, it receives a question in NL from the upper layer, translates it into a semantic query, launches it against the knowledge base, and finally returns the pertinent multimedia clips to the upper layer.

A. Domain Ontology

Our system can be more formally defined as a domain

ontology. With respect to the critics and proposals of the Semantic Web's layer architecture discussed in section III, we decided not to use a higher ontology language like OWL, but to use set- and graph theory to reason about the knowledge. Our layer architecture is based on a semantic model; a hierarchy of concepts (HC) [31].

Definition 1 (domain ontology): *A domain ontology O(L,H) is composed of a domain language L and a hierarchy of concepts H. L is the set of all words over a certain alphabet, $L \subseteq \Sigma^*$ that are known by the system for the given domain ontology. The hierarchical classification of concepts $H = (V, E, v_0)$ is a rooted tree, with V the set of nodes representing the concepts, E the set of edges and v_0 the root-node.*

Definition 2 (classification): *A multimedia clip d is classified under a concept v, if d is about v and there is not a more specific concept v' under which d could be classified.*

In certain cases, a document can be classified in more than one concept. For example the clip introducing the ARPA (the US agency, which was responsible for the invention of the ARPANet, the ancestor of the Internet) is classified in a concept named "Net" but also in a concept named "Inventor". Figure 4 shows an example of a taxonomy about computer history.

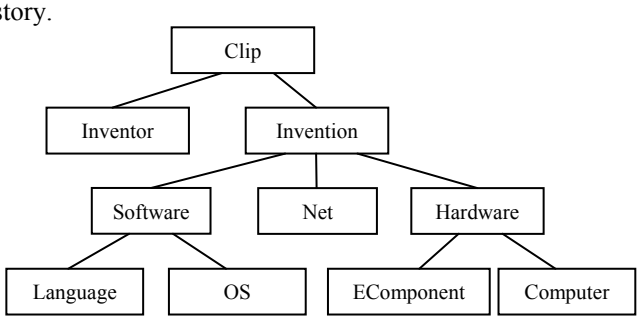

Fig. 4. Example of a taxonomy for computer history; a hierarchy of concepts (HC). Each node of this tree represents a specific concept. The clips are classified with respect to this HC.

B. Reasoning mechanism

It is not the topic of this paper to explain in detail how our inference engine works, see [5, 6] for details. But for a better understanding of our layer architecture, we will summarize its mechanism briefly.

As the user question is expressed in NL, but the knowledge base is described in RDF(S), a direct comparison is not possible because both representations are not compatible. Therefore, we transform the user question into a RDF semantic query that is launched against the knowledge base. This operation is done by the interpretation function $\mathcal{I}$.

Definition 3 (question interpretation): *The interpretation $\mathcal{I}$ of a user question q in NL for a domain ontology O and an allocation function g is written*

$$\mathcal{I}[q]_g^O = R$$

with R being the set of relevant documents.

The interpretation function $\mathcal{I}$ processes a user question q in three steps. Firstly, it filters out all the *noise* from the user question and keeps only the semantically relevant words. To do this, the inference engine uses as semantic knowledge source the domain language L, stored as domain dictionary (see definition 1). This dictionary could be an external knowledge source for example *WordNet* [32], or like in our implementation an adapted domain dictionary about computer history. The result is Φ, a set of semantically relevant objects and predicates. In a second step, this intermediate expression is mapped to a general assertion a_q using the allocation function g. In the third and final step, the mapped assertion is enriched with the values of the semantically important words from the user question q. This allows to create a semantic query which is launched against the knowledge base. The result is R, the set of relevant documents. Each document d comes with a quantifier σ that is expressed in a certain logic W. This allows to rank the documents from R according to their pertinence, if more documents were found. The choice of W depends on how expressive one wants to be in the approximation of the meaning of the concepts, and on the complexity of the NLP techniques used to process words. For example, W could be based on Bayesian Logic to compute the probability that d is relevant for a given user. An example of the reasoning mechanism is given in figure 5.

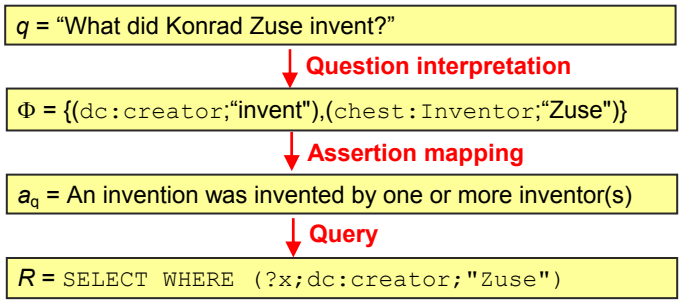

Fig. 5. Example of the interpretation function $\mathcal{I}$. A user question q expressed in NL is filtered to a set Φ of relevant words. The question is then mapped to a general assertion a_q and enriched with values from the original sentence. Finally, a semantic query is generated and executed.

C. Communication Interfaces

The Inference Layer can be seen as a black-box. It receives a NL question, performs some reasoning about it, and returns an answer. It communicates *via* standard interfaces; one to the upper Communication and Presentation Layers, and one to the lower Knowledge Layer. Both will be described below. But let us first recall that the inference engine must be platform independent, even portable, and accessible over a network by client applications from heterogeneous platforms (we refer here to clients as the user applications). Therefore, the modules of the Inference Layer were developed in Java; we used the Jena API [22].

The communication interface to the Knowledge Layer uses the RDF capabilities of Jena. The RDF(S) stores can be accessed either as a tree, either as a database. We used the

latter and the query language RDQL [23].

The communication interface to the Presentation Layer is based on two requirements. Firstly, the answer to the user's question must be available for the user in a very short time. This of course excludes sending the whole resulting multimedia clip(s) to the user. Instead, we transmit only two pieces of information per relevant document: the URI where the clip can be retrieved, and metadata about that document. It is up to the client application how this data will be presented to the user (see section VII). Secondly, the Inference Layer must be transparent to the user, independently if the inference engine runs as a process (service) on the user's local machine, or if it is accessed distantly as a web service. Hence, the answer of the Inference Layer must be encoded in a platform and system independent way. Our solution is to create a standard interface, based on a XML encoding, as it is done by most web services.

VI. COMMUNICATION LAYER

The Communication Layer allows a transparent communication between the client application (Presentation Layer) and the inference engine (Inference Layer). It should not be important if these two layers are on the same machine or not. Furthermore, the communication must be error-free, simple and hardware independent. It seemed to us that the best solution is a socket communication. A socket is basically a host identification and a port.

Using sockets has three advantages: it uses TCP/IP, offers an error-free transmission, and components for most development environments are widely available. As for the first issue, a socket communication is based on the TCP/IP protocol stack. Thus, TCP/IP must be installed on the user's computer to run the CHESt client application. The advantage is that TCP/IP is the most popular protocol at the moment and most people have it installed on their computer for using the Internet. Secondly, another advantage to use sockets is that the whole handshaking and error correction is assumed by the protocol stack. In fact, TCP/IP offers an error-free transmission. All details and used technologies (for example LAN adapter or analogous modem) are transparent for the user. Finally, all popular development environments offer components to easily implement a socket communication. This allows developers to create their own CHESt client application that communicates with the inference engine.

Technically, the Inference Layer is a service which runs on a distant or local machine. It listens on a specific port for a client call. A client call is the reception of a question string (the user question). The client holds the communication and waits for receiving the XML encoded answer. Then the transmission is ended. The client contacts the server by means of a port number, and an IP address or a hostname (*localhost* if the Inference Layer is located on the same machine).

VII. PRESENTATION LAYER

The Presentation Layer represents the interface between the user and the machine. It gets a question from the user and transmits it to the inference engine *via* the Communication Layer. In return, it displays the result(s) and allows the user to watch the clips. In our implementation, the Presentation Layer is available as web interface and as *Microsoft Windows* client application. Figure 6 shows a snapshot of our client application, which was developed in *Borland Delphi*. We begin this section with a discussion of the ergonomically and pedagogical aspects of the Presentation Layer, before presenting more technical details. Further pedagogical analysis about CHESt as e-learning tool can be found in [7].

A. Pedagogical Aspects

Let us put in evidence that the basic task of the Presentation Layer is to allow people, mostly not computer experts, to express their question by means of NL, and to watch the resulting document(s). Therefore, the graphical user interface (GUI) must be as simple and ergonomic as possible, especially because we are dealing with an educational tool. An e-learning interface should neither be too complicated nor too simple: if too complicated, the student gets lost in the menus; if too simple, (s)he could perceive the new tool as a game and risks not concentrating on the real issue of the lesson. The interface should be adapted to the needs of the user and keep him/her concentrated on what (s)he sees and learns. It is evident that all technical details (for example the complete communication with the lower layers, the reasoning and retrieval tasks, etc.) must be invisible for the user.

Let us mention here a possible improvement; an interface that adapts automatically to the user: simple interface for kids, more expressive interface for experts. Beside the pure layout problems, XSLT [26] could be used to filter too complicated documents from the resulting set.

B. Technical Aspects

The client application must be ready for use immediately, without installation or configuration procedures. For CHESt, the client application is one executable file that can simply be copied to the computer's disk. All needed components are embedded in the Presentation Layer: the communication interface for the socket communication, the XML parser for decoding the received answer from the inference engine, and the player for watching the clips.

Contrary to the Inference Layer, the Presentation Layer depends directly on the kind of documents in the knowledge base. In our case, we display multimedia clips, thus we must use a particular player which is supported by the operating system. This is the reason why the client application cannot be implemented in Java, which would of course be a promising solution; there would be only one platform independent client application. But as far as the authors know, no player for multimedia video sequences is available for Java. As described in section IV, the clips in our knowledge base are recorded as *RealMedia* files. These can be watched in any compatible player, for example the free *RealOne Player* [25]. We embedded this player as Active-X object [29] in all

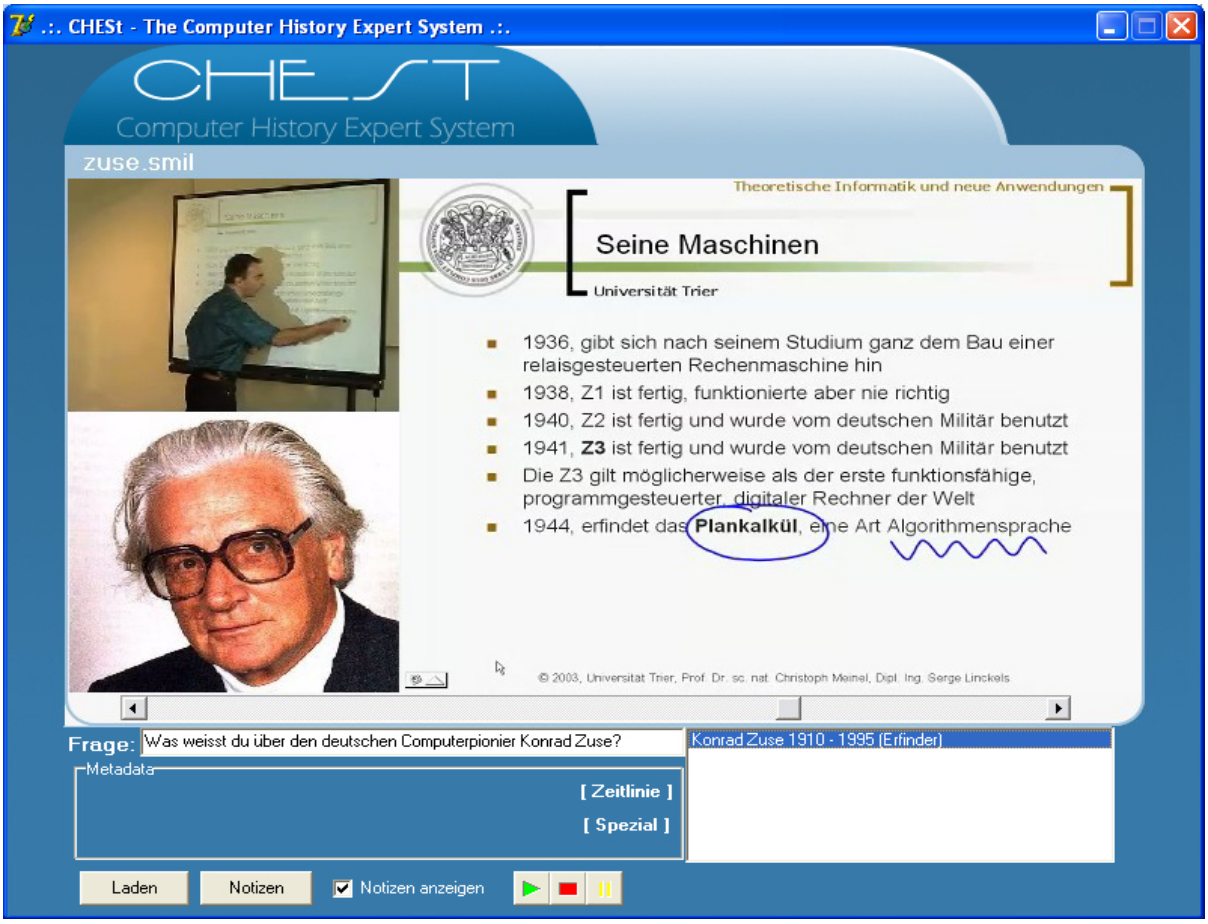

Fig. 6. Screenshot of the prototype CHESt with a semantic search about the inventor *Konrad Zuse*. The system does not return all clips about Zuse, but understands that the user wishes a general explanation about this inventor. The system displays the pertinent result(s) in the bottom right-hand corner, here only one result. Selecting an item will play the corresponding clip, like the one shown in this example, where the teacher uses an interactive board. Added handwritten comments made by the teacher are displayed in real time.

CHESt versions (see section VIII). This requires that the used player is installed on the user's computer. Specific players must be used for implementations on other platforms, for example the free *Helix Player* [27] for Linux.

The Presentation Layer must also be able to process the XML file that contains the encoded answer from the inference engine. Normally, most modern developing environments offer such XML parser components. CHESt displays the received list of clips, and also the delivered metadata. In some cases, the metadata can already be a satisfying answer for the user, so that the whole clip must not be watched.

At this place we want to report the negative result of another approach which we explored. The interaction with the inference engine could be seen as distant procedure call. A beautiful solution would be to use the *Java Native Interface* (JNI) to execute some Java code inside the client application. Most modern development environments offer a JNI; we used [24]. Unfortunately, not all Java programs are compatible with any JNI. We must admit that we were not able to develop a working solution for generating the XML encoded answer *via* a JNI-call. Therefore, we canceled this possibility and opted for the more straightforward socket solution.

VIII. CASE STUDY

In this section we start with some details about three different implementations that we tested for CHESt. We will explain the technical and pedagogical differences of each version. We will also describe some experiments with our prototype.

A. *Three Versions of CHESt*

In order to test the most appropriated configuration of our educational tool, we created three versions of CHESt:

(1) CHESt as pure web application and streaming of the clips.
(2) CHESt as pure local application and local access to the clips.
(3) CHESt as distributed application with streaming or local access to the clips.

Let us discuss some pedagogical and technical aspects of these three versions. (1) The first pure web version is the simplest to use; CHESt is available everywhere from a web browser. The Knowledge and Inference Layers can of course be on the same server. The only reasonable access to the clips is *via* a streaming server. (2) The second pure local version

has the ideal advantage, that no streaming server is necessary, thus performance problems are inexistent (no bottlenecks for the streaming server or for the network access). However, we see two main disadvantages. Firstly, the teacher must copy the client application, as well as the complete multimedia knowledge base on all used computers. This is time- and disk space consuming. Secondly, neither the clips, nor the software are prevented from being copied illegally. Nevertheless, it is an ideal solution for fast access to the clips without special technical infrastructure. We also managed to store the whole knowledge base with the required software on one single CD-ROM. (3) The third version of CHESt takes full profit of our semantic layer architecture, and implements a distributed system. It is the best adapted for being used in an intranet, for example inside a school. This is also the type of installation we kept for our further experiments (see below). In this version, the Knowledge Layer is located on a server, normally with a streaming access to the clips. The performance of the system is best if the servers that host the Knowledge- and the Presentation Layers are in the same LAN. A direct access to the clips on a shared disk is possible with the necessary grants. The Inference Layer is located on a different (or the same) application server. The students load the client application from an application server (or a local disk) and run it locally on their machine.

B. *Practical experiences*

CHESt was first used in the spring term of the academic year 2003/2004 in a school in Luxembourg/Europe. The aim was to test the interface and the multimedia content, not the searching algorithm. The first version of CHESt included a simple keyword search engine; the user entered some keywords, for example "transistor" and got a list of all clips in which this keyword appeared. He/she could then watch these clips. We were able to test how students get along with the interface and the multimedia content. We were particularly interested in how they would learn with such a tool and how motivating this tool was as opposed to classical learning methods.

Based on the results of that first experiment, we are currently preparing a second experiment with a version of CHESt, which fully implements our novel semantic layer architecture and our semantic search engine. This pilot project will be launched in the summer term 2005 in several selected schools in Luxembourg and Germany with a representative number of students. External psychologists and teachers will supervise and evaluate the test. The experiment will focus on two main issues: test the effectiveness of the semantic search engine and verify that school results can be improved if students use such an "intelligent" e-learning tool.

IX. RELATED WORK

In this section, we describe briefly some interesting and related projects. Some are related due to their layer architecture, others due to their reasoning- or retrieval mechanism, or NLP techniques.

In [2], an integration architecture is proposed which aims at exploiting data semantics in order to provide a coherent and meaningful (with respect to a given conceptual model) view of the integrated heterogeneous information sources. The architecture is split into five separate layers to assure modularization, providing description, requirements, and interfaces for each. It favors the on-demand retrieval paradigm over the data warehousing approach. In general, there are two paradigms for knowledge based systems: the data warehouse approach and the on-demand retrieval approach. The first builds on a central data collection which can be queried. The second collects the data from different sources in respect to a user question. The novelty of the proposed architecture lies in the combination of semantic and on-demand driven retrieval.

The KIM [29] platform provides a novel *Knowledge and Information Management* (KIM) infrastructure and services for automatic semantic annotation, indexing, and retrieval of documents. It provides mature infrastructure for scaleable and customizable information extraction as well as annotation and document management, based on GATE (http://gate.ac.uk/). In order to allow easy bootstrapping of applications, KIM is equipped with an upper-level ontology and a knowledge base providing exhaustive coverage of entities of general importance. The ontologies and knowledge bases involved are handled using cutting edge Semantic Web technology and standards, including RDF(S) repositories, ontology middleware and reasoning. From a technical point of view, the platform allows KIM-based applications to use it for automatic semantic annotation, content retrieval based on semantic restrictions, and querying and modifying the underlying ontologies and knowledge bases.

A very interesting approach for querying a distributed knowledge source like the WWW and to automatically rank the resulting documents was proposed by the university Blaise Pascal Clermont2 and the university Claude Bernard Lyon1 [33]. The user can express his question in NL. Description Logics were used as a formal representation language for specifying documents and queries. The matching step consists in comparing the two terminologies obtained from a query and a document. Given two terminologies $\mathcal{T}_Q$ and $\mathcal{T}_D$ describing a query Q and a document D respectively, the goal is to find the elements in $\mathcal{T}_Q$ and $\mathcal{T}_D$ that match. This is done by a matching function that takes two terminologies as input and produces a one-to-one mapping between defined concepts of the two terminologies that correspond semantically to each other. Finally, the documents are ranked according to the size of the extra information contained in the query and not in the documents. The extra information is calculated with the help of the difference operation between pairs of mapped elements. The proposed algorithm computes the difference between $\mathcal{ALE}$-concept descriptions. It takes into account linguistic relations (synonymy, hyponymy...) between concept names occurring in the two descriptions.

X. CONCLUSION AND OUTLOOK

We presented in this paper a semantic layer architecture for an educational tool concerning computer history. The essential advantage of our semantic layer architecture, especially in an educational environment is that the complexity of the reasoning mechanism as well as all underlying technical representations of the knowledge, are transparent to the user. It also respects a distributed approach; the knowledge base, the inference engine and the client application can be on different machines and sites. The inference engine is implemented in a platform independent layer. The access to the documents in the knowledge base can be configured with respect to the available infrastructure and resource constraints. Our layer architecture was implemented in a prototype called CHESt, but it can be applied easily to any other system. The facility of interacting with the tool and the multimedia aspect of the answers returned by the system make CHESt a useful complement to traditional education.

Our proposed layer architecture uses technologies from the Semantic Web, and is based on the original Semantic Web layer architecture, but it respects recent proposals and critiques. Unfortunately, the missing of higher layers in our pragmatic solution implies that no proof or guarantee can be given to the user that the proposed clips are really the most pertinent.

We are currently working on the improvement of the reasoning task, in order to create a more deterministic inference engine, maybe by adding a more powerful layer on top of our Inference Layer. Furthermore, a more expressive formalism for reasoning about the knowledge seems to be necessary. We are about to investigate whether to take the approach that is proposed in [3] (see section III), or whether to use the W3C approach by using the ontology language OWL with its OWL-DL extension, based on Description Logics.

REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web". *Scientific American*, 2001.

[2] R. Vdovjak, and G.J. Houben, "RDF based architecture for semantic integration of heterogeneous information sources". *Proc. International Workshop on Information Integration on the Web (WIIW)*, Rio de Janeiro, Brazil, 2001, pp. 51-57.

[3] B. C. Grau, "A possible simplification of the semantic web architecture". *Proc. ACM World Wide Web (WWW)*, New York, USA, 2004, pp. 704-713.

[4] E. Miller, "W3C Track - The Semantic Web", *ACM World Wide Web (WWW)*, http://www.w3.org/2002/Talks/www2002-w3ct-swintro-em/, Honolulu, Hawaii, USA, 2002.

[5] S. Linckels, and Ch. Meinel, "An application of semantics for an educational tool". *Proc. International Association for Development of the Information Society Applied Computing (IADIS AC)*, Lisbon, Portugal 2004, pages II 234-II 239.

[6] S. Linckels, and Ch. Meinel, "Automatic interpretation of natural language for a multimedia e-learning tool". *Proc. International Conference on Web Engineering (ICWE)*, Munich, Germany, 2004, pp. 435-439.

[7] S. Linckels, and Ch. Meinel, "An educational tool that understands students' questions". *Proc. Association for Educational Communications and Technology (AECT) "All That Jazz"*, Chicago, Illinois, USA, 2004.

[8] P. F. Patel-Schneider, and D. Fensel, "Layering the Semantic Web: problems and directions". *Proc. International Semantic Web Conference (ISWC)*, Sardinia, Italia, 2002.

[9] I. Horrocks, P. F. Patel-Schneider, and F. v. Harmelen, "From SHIQ and RDF to OWL: The making of a web ontology language". *Journal of Web Semantics*, 1(1):7-26, 2003.

[10] J. Pan, and I. Horrocks, "RDFS(FA): A DL-ised sub-language of RDFS". *Proc. Description Logic Workshop*, volume 81 of CEUR (http://ceur-ws.org/), pages 95-102, 2003.

[11] World Wide Web Consortium, "Resource Description Framework (RDF) / W3C Semantic Web", http://www.w3.org/RDF/.

[12] World Wide Web Consortium, "Web Ontology Language OWL / W3C Semantic Web Activity", http://www.w3.org/2004/OWL/.

[13] M. Ma, V. Schillings, T. Chen, and Ch. Meinel, "T-Cube, a multimedia authoring system for eLearning". *Proc. Association for the Advancement of Computing in Education (AACE)*, Phoenix, USA, 2004, pp. 2289-2296.

[14] Ch. Meinel, and V. Schillings, "tele-TASK - Teleteaching Anywhere Solution Kit". *Proc. ACM Special Interest Group on University and College Computing Services (SIGUCCS)*, Providence, USA, 2002, pp. 130-133.

[15] T. Chen, M. Ma, Ch. Meinel, and V. Schillings, "Tele-TASK, Teleteaching Anywhere Solution Kit", http://www.tele-task.de/.

[16] S. Powers, *Practical RDF, Solving Problems with the Resource Description Framework*, O'Reilly, USA, 2003.

[17] Dublin Core Metadata Initiative (DCMI), http://dublincore.org.

[18] World Wide Web Consortium, "Representing vCard Objects in RDF/XML", http://www.w3.org/TR/2001/NOTE-vcard-rdf-20010222/.

[19] S. Linckels, CHESt NS, http://www.linckels.lu/chest/elements/1.1/.

[20] R. Troncy, "Integrating structure and semantics into audio-visual documents". *Proc. International Semantic Web Conference (ISWC)*, Sanibel Island, USA, 2003.

[21] A. Kiryakov et al., "Semantic annotation, indexing, and retrieval". *Proc. International Semantic Web Conference (ISWC)*, Sanibel Island, USA, 2003.

[22] HP Labs Semantic Web Research, " Jena – a Semantic Web framework for Java", http://www.hpl.hp.com/semweb/.

[23] L. Miller, A. Seaborne, and A. Reggiori, "Three implementations of SquishQL, a simple RDF query language". *Proc. International Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002.

[24] M. Mead, "Using the Java native interface with Delphi", http://www.pacifier.com/~mmead/jni/delphi/.

[25] RealNetworks, "RealOne Player", http://www.real.com/.

[26] World Wide Web Consortium, "XSL Transformations (XSLT)", http://www.w3.org/TR/xslt/.

[27] Helix Community, https://helixcommunity.org/.

[28] RealNetworks, "RealOne Player scripting guide", http://service.real.com/help/library/guides/realonescripting/browse/htmfiles/title.htm/.

[29] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov, " KIM – semantic annotation platform". *Proc. International Semantic Web Conference (ISWC)*, Sanibel Island, USA, 2003.

[30] R. Baeza-Yates, B. Ribeiro-Neto, *Modern Information Retrieval*, Addison-Wesley, USA, 1999.

[31] P. Bouquet et al., "Semantic coordination: a new approach and an application". *Proc. International Semantic Web Conference (ISWC)*, Sanibel Island, USA, 2003.

[32] WordNet, http://www.cogsci.princeton.edu/~wn/.

[33] N. Karam et al., "Semantic matching of natural language web queries". Proc. *International Conference on Web Engineering (ICWE)*, Munich, Germany, 2004, pp. 416-429.