

Model-driven development of personalized, multichannel interfaces for audiovisual search: the PHAROS approach

Piero Fraternali¹, Alessandro Bozzon¹, Marco Brambilla¹, Vincenzo Croce², Kathrine Hammervold³, Eric Moore³, Francesco Saverio Nucci², Michel Plu⁴

¹Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy; ²Engineering SpA R&D Lab, Rome, Italy
³Fast, A Microsoft Subsidiary, Oslo, Norway; ⁴France Télécom R&D, Lannion, France

E-mail: ¹{piero.fraternali,alessandro.bozzon, marco.brambilla}@polimi.it,
²{vincenzo.croce,francesco.nucci}@eng.it, ³{kathamm,eric.moore}@microsoft.com,
⁴michel.plu@orange-ftgroup.com

Abstract: In this paper we illustrate the model-driven development approach applied to the user interface of an audiovisual search application, within the European project PHAROS. We show how conceptual modelling can capture the most complex features of an audio-visual Web search portal, which allows users to pose advanced queries over multimedia materials, access results of queries using multimodal and multi-channel interfaces, and customize the search experience by saving queries of interest in a personal profile, so that they can be exploited for asynchronous notification of new relevant audiovisual information. We show how model-driven development can help the generation of the code for sophisticated Rich Internet Application front-ends, typical of the multimedia portals of the future.

Keywords: model-driven development, audiovisual search engines, multi-channel user interfaces, personalization

1 INTRODUCTION

In* the past few years we have witnessed a tremendous growth in the amount of available digital data, both in a textual and multimedia format: a study of University of California [17] suggests that 3.4 billion terabytes of new information are created each year. The lowest estimates translate this into a 30 percent annual growth in new data stored in paper, film, magnetic and optical medium. At the same time, *search* has become the primary way of interacting with data and by the end of 2008 more than 50% of applications are predicted to include a search facility as a primary interface for end users [4]. In this context, Web applications like multimedia search portals are bound to become the new media channels. Depending on the targeted application domain, the indexing and retrieval processes have to be tailored to a wide spectrum of functional and non functional requirements. Focusing on the front-end of the application and on user-interaction, the challenges to

application developers are related to the complexity deriving from:

- the range of possible query modalities that can be offered to the user: for example, an audio-visual search engine might offer simple keyword-based query functionalities, metadata search, similarity search, geographical search, or time-based search [12];
- the need for ubiquitous, multi-channel, multi-device access; for example, a user may start a query-by-example session while roaming with his mobile terminal and then refine or expand the results at home with a faceted search using a PC interface;
- the huge amount of available data, which leads to an information overload that can be overcome only by means of multiple interaction patterns, like active searching, monitoring, browsing and proactive notification [3];
- the presence of many supplementary functionalities, like tagging, commenting or, in general, everything related to the social aspects of a multimedia search portal;
- the architectural distribution of the components (front-end, back-end, data repositories), which can be required in order to support business or performance requirements.

Such advanced audiovisual search functionalities should be deployed not only within dedicated search portals, but also flexibly integrated within existing systems, for example to add sophisticated knowledge discovery capabilities to applications supporting the daily business processes of an enterprise. Therefore the challenge arises of defining suitable architectural frameworks and development processes for enabling the seamless integration of audiovisual search into modern Web-enabled solutions: Web content publishing and manipulation, Web Service publication, discovery, and invocation, user profiling, interface adaptation and personalization. These requirements, coupled to the

* This work is partially funded by the European Union Sixth Framework Programme (FP6), through the integrated project Pharos (IST-2006-045035).

reduced time-to-market of Internet-time applications, demand for novel methodologies and tools supporting openness, extensibility, and fast prototyping. We claim that such needs can be fulfilled by providing instruments that abstract from implementation details and that represent the final system at a higher level, using appropriate conceptual models of the application at hand.

The work described in this paper is part of the activities conducted within the PHAROS project (Platform for searchIng of Audiovisual Resources across Online Spaces), an Integrated Project co-financed by the European Union under the Information Society Technologies Programme. PHAROS, which comprises 13 partners from 9 different countries, aims at *“developing an innovative audiovisual technology platform ... that will take user and search requirements as key design principles and which will be deeply integrated with user and context technologies...”* [4].

In this paper, we describe the application of the Web Modelling Language (WebML [6]) and of the WebRatio toolsuite [7] to the development of the front-end of the PHAROS audiovisual search platform.

2 PHAROS ARCHITECTURE

Prior to focusing on the model driven development of the PHAROS front-end, we illustrate the rationale and main design principles behind the PHAROS architecture.

The distinctive aspect of the PHAROS platform is its being an open framework for developing audiovisual search solutions, rather than a closed information retrieval system. This means that every functionality of the architecture is conceived to be pluggable: the modules in charge of registering the content, the feature extraction and content annotation algorithms, the workflow manager of the annotation process, the user and community management modules are all designed for being flexibly added to the core framework of the architecture. Openness is also applied w.r.t. interactions with external actors, typically content providers, implementing customized platform extensions for content publishing. The core element of PHAROS consists of an XML search engine, which operates on the annotations extracted from the content processing workflow, expressed in the Audio Visual RSS (AVRSS) format [1], an XML vocabulary extending the MPEG7 media description format [10].

Figure 1 depicts a high level view of the PHAROS platform, highlighting its main components and their functionalities: all the modules in the architecture are designed according to the SOA paradigm, thus enhancing functional distribution while leveraging on Web service for message exchanges.

A fundamental issue in the design of PHAROS is concerned with the definition of the graphical user interface, which should adequately support the variety of users experiences that show up in audiovisual search session. To validate the design, several showcase applications have been defined, built on the top of a

distributed PHAROS platform instance, to be used for gathering user’s feedback and for distilling best practices in the various dimensions of audiovisual search.

The level of flexibility of the envisioned interfaces has prompted for an approach to development centred around frequent cycles of fast prototyping, feedback and code adaptation. We have chosen to exploit a model-driven approach to development, which can help in representing the features of the front-end in a technology-independent way and in generating the code automatically for different technologies and usage scenarios, like, e.g., pure HTML PC-based interfaces, mobile terminal interfaces, Rich Internet Applications with AJAX and/or FLASH, and more.

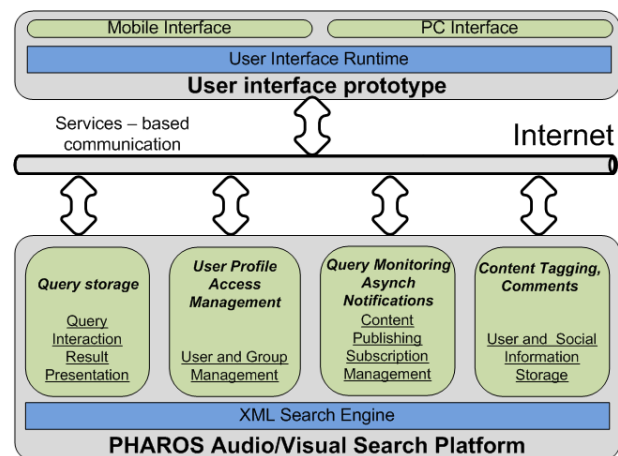


Figure 1: The PHAROS platform architecture

In particular, we have focused on the modelling and automatic generation of:

- multi-channel, and multi-modal user interfaces;
- advanced query composition for multi-media materials;
- advanced features for accessing the query results through
- sophisticated Rich Internet Application interfaces for accessing the query results;
- advanced personalization features.

As a development methodology and toolset, we have exploited the Web Modelling Language (WebML) [6], a Domain Specific Language for the specification of the content, business logics, hypertext, and presentation of a Web application, and WebRatio [7], a CASE tool that supports WebML-based application design and provides automatic code generation for the JEE platform and for a broad range of interface presentation technologies.

3 WEBML BACKGROUND

WebML[6] is a visual language originally conceived for supporting the design and implementation of so-called data intensive Web applications, defined as Web sites for accessing and maintaining large amounts of structured data, typically stored in a database management system. The core of the WebML language is based on three main conceptual models: the data model, specifying the

application data, the hypertext model, addressing the specification of the set of links, pages and content that form the front-end of the application, and the presentation model, specifying the look&feel and the layout of the Web pages. The definition of the language has then evolved towards more complex classes of applications, including process-driven applications and Web Service orchestrations, to enable the modelling and implementation of distributed applications for multi-actor workflow enactment [5].

In the design of the PHAROS front-end, we exploited the Web services modelling capability of WebML to express the integration of the front-end of the application with the services exposed by the PHAROS search platform. More recently, WebML has been extended towards Rich Internet Applications (RIAs) [2], to cope with the requirements of the so-called Web 2.0. RIAs provide powerful interfaces for representing complex processes and data, while minimizing client-server data transfers and moving the interaction and presentation layers from the server to the client. In the design, we exploited some distinguishing features, like partial page refreshment (i.e., only the subparts of the page that are affected by the user interaction are updated), to improve the user experience in the interface of a search engine. Moreover, the native support of RIA technologies to multimedia contents like video and audio has been also put to work. The combination of all such extensions of the WebML language are at the basis of the conceptual specification of the PHAROS showcase, as we will see through a case study in the next section.

4 THE PHAROS USER INTERFACE

The showcase Web application described in this paper consists of two interfaces: one for standard PCs and one for mobile terminals. The developed uses cases cover the following scenarios:

- *Keyword search and query refinement on locally stored audiovisual content:* a user with a PC exploits an HTML+AJAX interface to submit keyword-based queries over a collection of video materials that have been registered into an instance of the PHAROS platform. Query results are presented together with metadata extracted from the audiovisual data. The retrieved metadata are fed into a faceted-search panel, whereby the user can refine the query and open the video matching his query, directly accessing the relevant scenes.
- *Similarity query with a mobile terminal and subsequent registration of the query in the user's profile:* a user records a fragment of a song with his cellular phone and sends a fingerprint to the PHAROS back-end, receiving back the metadata about the matching song. He can then save the query on the server, refine it later on his PC, and receive multi-channel notifications when new content is added to the PHAROS repository that matches the saved query.

The interface exploits both the natively available information (e.g., appearing in the page that host the content, like title, description, or production date) and a set of annotations automatically extracted by the PHAROS platform during the indexing process; examples of annotations are: speech-to-text transcriptions, the speaker's gender and name, the music mood and rhythm, or the concept represented in a keyframe of the video (e.g., snow, sky, and so on).

The interaction of the PHAROS front-end and back-end addresses two main issues: on the one side, the front-end manages the query building process; on the other side, the results produced by the search engine are reorganized at the proper level of granularity and rendered in the user interface.

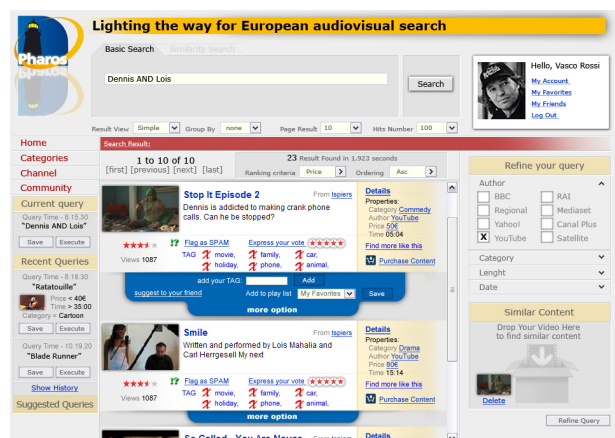


Figure 2: The PHAROS front-end for the keyword search of audiovisual data. On the right hand side, the query refinement panel exploits the extracted annotations for the current video.

A special concern of audiovisual result presentation has to do with time, because annotations may contain the temporal references where the desired query parameters occur, for both audio and video files. Then, audiovisual content must be shown using different layout templates and, for each result matching a given query, the user interface must represent the temporal segments of occurrence for the computed matches. For example, when searching for *Sarkozy* inside a video collection, the PHAROS platform will return, for all the matching videos, the temporal segments where a speaker pronounced the name of the French president. The interface allows users to be automatically directed to the exact point in the video where the match has been found. Conversely, when a user plays a video, the list of matching annotations are shown in synch with the video execution.

Annotations are represented across the tiers of the PHAROS architecture as XML documents compliant with the AVRSS specification [1]; user's queries are encoded as statements in the XPath and MP7QF query languages. The application front-end translates the user's interactions into the needed query expressions and submits them to the query engine; currently, the automatic generation of queries is supported for (i) keyword-based queries, (ii) annotation-based queries and filtering, and (iii) similarity-based search on audio

contents. Queries can be iteratively built as an arbitrary combination of such options. Example of supported queries are: a) find all the videos related to tourism in Bavaria, b) find the videos talking about Sarkozy and where a speaker is Rachida Dati, c) find Shakira party music videos similar to a given song. Additionally, the showcase implements collateral functionalities like the management of user-generated tags for the published contents, the possibility of storing queries for later reuse, and of marking queries as monitored, in order to receive SMS and e-mail notifications when new contents matching the given query are published.

In the next sections we show extracts of the conceptual models designed for the described PHAROS showcase. The complete data and hypertext models for both PC and mobile interfaces, as well as an extended presentation relative to the showcase applications are available in the project's web site: <http://www.pharos-audiovisual-search.eu>

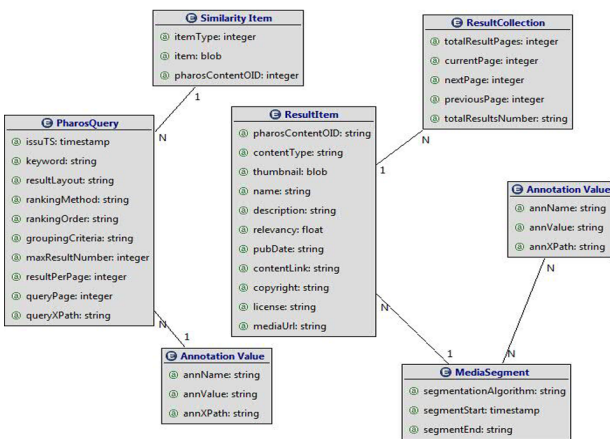


Figure 3: excerpt of the data model representing the PHAROS front-end

4.1 The data model for query composition and result presentation

Figure 3 shows a simplified data model of the PHAROS showcase content. The query building process is managed by means of the *PharosQuery*, *AnnotationValue*, and *SimilarityItem* entities, which, respectively, allow the modelling of basic search parameters (e.g., keywords, number of results for page, etc.), advanced search constraints based on contents' annotations, and similarity criteria for content-based search. The results presentation process exploits the definition of a *ResultCollection* entity, which represents general information about the query result (e.g., the number of retrieved results). Entity *ResultItem*, contains information about each retrieved result, like the unique pharosContentID, its thumbnail representation, and so on. For each result item, its temporal decomposition is represented by means of the *MediaSegment* entity, which, for a given decomposition algorithm (e.g., based on shot detection) provides data about the starting end ending time of the given segment. Finally, the *AnnotationValue* entity models the actual annotations, as

produced by the indexing process performed at the back-end the PHAROS platform.

4.2 The model of the interface and of the integration with the back-end services

In WebML, one or more front-ends can be specified on top of a data model, expressed as *site views*, that is, diagrams that represent the composition and navigation of the hypertext interface. Figure 4 shows a (simplified) site view, representing the PC-based front-end that supports the user in performing the query composition and result presentation processes. The WebML notation essentially consists of *containers* (e.g., pages and operation sequences, denoted as named rectangles) and *components* (called *units*), denoted by labelled icons. Components are of two kinds: *content units*, which are placed inside pages and produce content displayed in the interface; *operation units*, which are placed outside pages and denote arbitrary business actions triggered by the interaction of the user. Content and operation units are connected by *links*, denoted as arrows, which represent both parameter passing dependencies between components and interaction devices (e.g., navigable hyperlinks, submit buttons, etc).

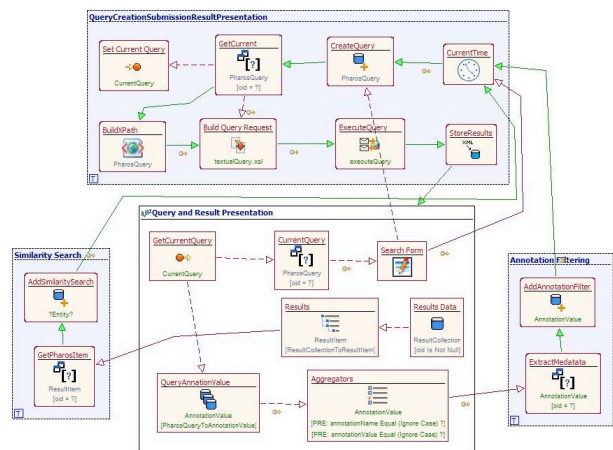


Figure 4: excerpt of the hypertext model (site view) representing the PHAROS front-end

The site view diagram illustrated in Figure 4 contains a single Web page (*Query and Result Presentation*), and by three sequences of operations, each one dedicated to specific query processing sub-task (*QueryCreationSubmissionResultPresentation*, *SimilaritySearch*, *AnnotationFiltering*).

The *Query and Result Presentation* page supports the formulation of keyword-based query; the *Search Form* unit allows the user to insert the desired keywords; its outgoing link triggers an operation chain, which first creates a new query (*CreateQuery* unit), then, transforms the E-R representation of the query into an XQuery expression, by means of the *BuildXQuery* unit; next, a SOAP request is created (*Build Query Request*) in order to invoke the search Web service exposed by the PHAROS platform (*Execute Query*). Finally, the response XML file containing the matching results of the query is transformed and stored in the *ResultCollection* entity, by means of the *StoreResults* unit. The operation

chain leads back to the *Query and Result Presentation* page, where the user is presented with the list of matching results, by means of the *Result List* hierarchical index unit. For every element in the result list, its metadata are shown along with information about the temporal segment and type (audio segment or video segment) for the query match. By clicking on a temporal segment timestamp, the user is directed to a separate page (not shown in Figure 4), which contains all the details about the result element and starts the play out of the associated video from the selected timeframe. Additionally, the *Query and Result Presentation* page comprises the *Aggregator* multichoice unit, which allows query refinement based on the annotation values retrieved from the result list: by selecting from the multichoice index the desired filtering conditions, the query composition process enriches the query with new operators (the *Annotation Filtering* operation chain) and re-executes the expanded query.

4.3 Advanced query capabilities

Besides the basic query construction and result presentation functions described in the previous paragraph, the front-end supports also more advanced capabilities.

4.3.1 Similarity search

The hypertext model of Figure 4 allows the execution of similarity search, too. Any item displayed in the *Results* content unit can be dragged and dropped onto the query refinement page area that displays the annotations for refining the query (denoted by the *Aggregators* multichoice index unit); this action triggers the *Similarity Search* operation sequence, which identifies similar items (with the *GetPharosItem* operation unit) and adds similarity conditions to the refined query (with the *AddSimilaritySearch* operation unit), and then starts the operation sequence for executing the expanded query.

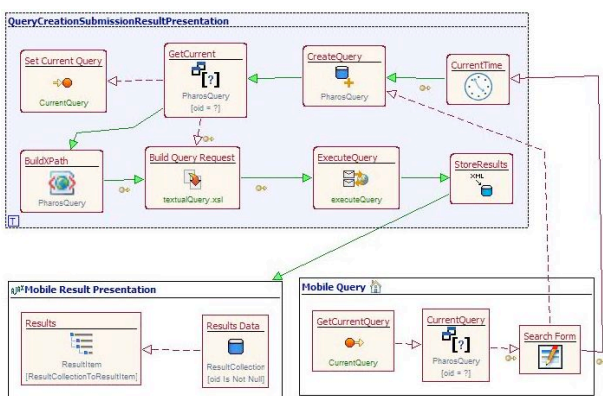


Figure 5: The WebML site view for the keyword search of audiovisual data with a mobile terminal. Functionality is split into two separate pages with simpler structure and navigation.

4.3.2 Multi-channel interfaces

The page shown in Figure 2 is the rendition of the *Query and Result Presentation* page model of Figure 4 for PC-based access. Thanks to the conceptual modelling

approach, it is possible to reuse the model of the operations for query preparation, execution and refinement, and embed them into a site view specifically tailored to the visualization and interaction capability of small-screen mobile terminals.

This simply requires extending the model with an additional site view, whose pages represent the organization of the query interface most suitable for the mobile access devices of interest.

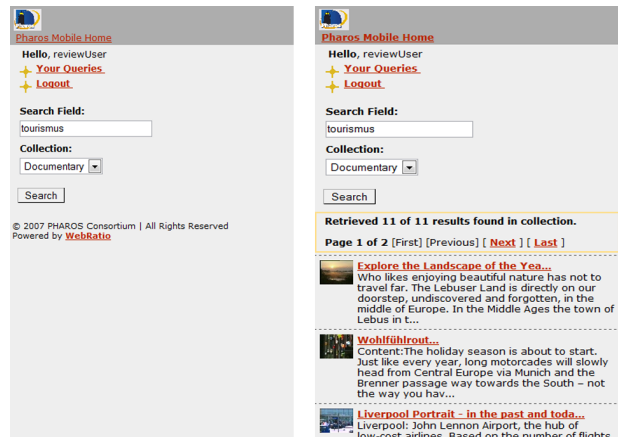


Figure 6: The rendition of the pages modelled in Figure 5.

Figure 5 and Figure 6 respectively depict the WebML hypertext model and the two corresponding Web pages that have been designed for the mobile access scenario, where just basic keyword searches and result browsing are supported; to cope with the limited interaction and rendering capabilities of mobile devices, the offered functionalities are split into two separate pages, leveraging simpler structure and navigations to ease the searching task.

4.4 Implementation notes

The WebML model of the PHAROS front-end has been implemented using the automatic code generation functionality of WebRatio [7]. This tool produces the complete implementation code corresponding to a given WebML model, including business logic artefacts (e.g., Java Beans, action classes for the Struts framework [15], controller configuration files, etc), presentation artefacts (JSP pages, JavaScript code, etc), data access artefacts (SQL queries, Enterprise Java Beans, Hibernate configuration files [16], etc), and Web Service artefacts (SOAP message composers, XML-XML and XML-relational transformations, WSDL descriptors, etc).

A quite unique feature of WebRatio is the possibility to incorporate into the code generator arbitrary presentation rules for producing virtually any look & feel and page organization.

We have exploited this possibility heavily, and extended the standard HTML-based Web code generator with the support of advanced AJAX interaction patterns (drag&drop, asynchronous Web server requests, visual transitions), to achieve a compact and easy to use interface for the quite complex task of constructing and refining audiovisual queries.

5 RELATED WORK

Audiovisual search is subject to many research efforts, from multiple perspectives: content processing and annotation for audio and video [10], architecture of the search engine [11][13][14], annotation representation formats [1][10], scalability, distribution and performance [18]. The focus of our work is less explored: how conceptual modelling and code generation techniques, emerged in the Web engineering field [9], can be brought to bear on the problem of integrating audiovisual search functions in the development of distributed, data-intensive Web applications. Even if several methodologies and tools exist for the conceptual modelling of both general-purpose and vertical Web applications (e.g., collaborative web applications [8], service-oriented and work-flow driven web applications [5]), to our knowledge there is no previous work exploring how to address the requirements of audiovisual search interfaces using conceptual modelling and automatic code generation. The original contribution of our work is a set of implemented showcase applications that demonstrate how to seamlessly integrate the data model of personalized refineable queries, the advanced query construction and result presentation front-end interfaces, and the SOA architecture wrapping a distributed audiovisual search engine.

6 CONCLUSIONS AND FUTURE WORK

In this paper we have discussed the preliminary experience of implementing the front-end interfaces of the PHAROS audiovisual query platform using an approach based on conceptual modelling and on code generation. The major benefit is the orthogonal representation of the data model, Web service operations, and interface composition and navigation, which contribute to making up the front end of the audiovisual search system.

Our future work will concentrate on extending the set of requirements that can be modelled and automatically implemented with the illustrated approach. In particular, we will enrich the range of query and result fruition modalities offered to the user (e.g., by adding geographical queries, temporal queries, and more sophisticated ways of specifying similarity search), and address also the modelling of the management interfaces needed for orchestrating the content acquisition and indexing processes, which may require quite sophisticated workflows in advanced audiovisual search applications. Finally, a work aimed at supporting proactive context-sensitive searches on mobile devices has been undertaken in order to provide the PHAROS Platform with Web-on-the-Move features.

Acknowledgements

We wish to thank all the researchers who contributed to the specification, design and implementation of the PHAROS platform, as well as the representatives of the user groups that performed the evaluation of the designed interfaces.

References

- [1] Av-rss. <http://www.pharos-audiovisual-search.eu/avrss/avrsexample.xml>
- [2] A. Bozzon, S. Comai, P. Fraternali, G. T. Carughi: Conceptual modeling and code generation for rich internet applications. ICWE 2006;
- [3] M. Bates. Toward an integrated model of information seeking and searching. *The New Review of Information Behaviour Research*, 3:1–15, 2002.
- [4] S. DeBald, W. Nejdil, F. Nucci, R. Paiu, and M. Plu. PHAROS platform for search of audiovisual resources across online spaces. In SAMT2006, 2006.
- [5] I. Manolescu, M. Brambilla, S. Ceri, S. Comai, P. Fraternali: Model-driven design and deployment of service-enabled Web applications. *ACM Trans. Internet Techn.* 5(3): 439-479 (2005)
- [6] P. Fraternali, S. Ceri et al. *Designing Data-Intensive Web Applications*. Morgan Kaufmann Publishers Inc., 2002.
- [7] Webratio. <http://www.webratio.com>
- [8] M. Matera, A. Maurino, S. Ceri, P. Fraternali: Model-driven design of collaborative Web applications. *Softw., Pract. Exper.* 33(8): 701-732 (2003)
- [9] Athula Ginige, San Murugesan: Guest Editors' Introduction: The Essence of Web Engineering-Managing the Diversity and Complexity of Web Application Development. *IEEE MultiMedia* 8(2): 22-25 (2001)
- [10] Hyoung-Gook Kim, Nicolas Moreau, and Thomas Sikora: MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval. Wiley & Sons, October 2005
- [11] Ross S. Purves et al.: The design and implementation of SPIRIT: a spatially aware search engine for information retrieval on the Internet. *International Journal of Geographical Information Science* 21(7): 717-745 (2007)
- [12] Dian Tjondronegoro and Amanda Spink: Web search engine multimedia functionality. *Information Processing & Management*, Volume 44, Issue 1, January 2008, Pages 340-357.
- [13] Silvia Pfeiffer, Conrad Parker, André Pang: The Continuous Media Web: a distributed multimedia information retrieval architecture extending the World Wide Web. *Multimedia Syst.* 10(6): 544-558 (2005)
- [14] Ponnada, Sharda: Model of a Semantic Web Search Engine for Multimedia Content Retrieval. *ICIS*, pp. 818-823, 6th IEEE/ACIS ICIS 2007
- [15] Struts framework, <http://struts.apache.org/>
- [16] Hibernate relation persistence framework, <http://www.hibernate.org/>
- [17] P. Lyman and Hal R. Varian: *How Much Information*, 2003. Retrieved from <http://www.sims.berkeley.edu/how-much-info-2003>
- [18] Moreira, J. E., Michael, M. M., Da Silva, D., Shiloach, D., Dube, P., and Zhang, L. 2007. *Scalability of the Nutch search engine*. In Proceedings of the 21st Annual international Conference on Supercomputing (Seattle, Washington, June 17 - 21, 2007). ICS '07. ACM, New York, NY, 3-12.
- [19] Maged Michael, Jose E. Moreira, Doron Shiloach, Robert W. Wisniewski.: *Scale-up x Scale-out: A Case Study using Nutch/Lucene*, ipdps, p. 441, 2007 IEEE International Parallel and Distributed Processing Symposium, 2007
- [20] Fagni, T., Perego, R., Silvestri, F., and Orlando, S: *Boosting the performance of Web search engines: Caching and prefetching query results by exploiting historical usage data*. *ACM Trans. Inf. Syst.* 24, 1 (Jan. 2006), 51-78.
- [21] Gruhne, M., Tous, R., Delgado, J., Doeller, M., and Kosch, H. 2007. MP7QF: An MPEG-7 Query Format. In *Proceedings of the Third international Conference on Automated Production of Cross Media Content For Multi-Channel Distribution* (November 28 - 30, 2007). AXMEDIS. IEEE Computer Society, Washington, DC, 15-18.