



COVER SHEET

This is the author-version of article published as:

Joly, Adrien and Tjondronegoro, Dian (2006) An Adaptive and Extensible Web-based Interface System for Interactive Video Contents Browsing. In *Proceedings International Joint Conferences, E-conference*.

Copyright 2006 (please consult author)

Accessed from <http://eprints.qut.edu.au>

An Adaptive and Extensible Web-based Interface System for Interactive Video Contents Browsing

Adrien Joly

Dian Tjondronegoro

Faculty of Information Technology,
Queensland University of Technology
GPO Box 2434, Brisbane 4001, Queensland

adrien.joly@gmail.com, dian@qut.edu.au

Abstract - With the growing popularity of mobile devices (including phones and portable media players) and coverage of Internet access, we tend to develop the need of consuming video content on the move. Some technologies already allow end-users to watch TV and listen to news *podcasts* or download music videos on their devices. However, such services are restricted to a provider's selection of pre-formatted and linear content streams. Hence, we propose a web-based interface system that supports interactive contents navigation, making it possible for end-users to "surf" on video content like they are used to on the Web. This system is extensible to any specific domain of video contents, any web-enabled platform, and to any browsing scheme. In this paper, we will explain the architecture and design of this system, propose an application for soccer videos and present the results of its user evaluation.

Keywords: architecture, multimedia, content, browse, hypermedia, navigation, video, semantic, XML, pipeline

I. INTRODUCTION

Nowadays, most of us carry latest-technology mobile devices, such as mobile phones, PDA, pocket game consoles and portable media players, allowing to play video contents wherever we go. Along with the increasing coverage and bandwidth of wireless Internet access, today's consumers expect a richer, easier and more rewarding experience from their video-enabled devices to find, select, retrieve and consume video content on the move. The popular solution of cellular network providers is to propose a restrictive range of videos to download and/or access to certain TV channels. But this approach is very restrictive as these providers "push" their own content instead of leaving the consumer browse any content from any source on the Internet.

While streaming any video from the Internet is becoming possible on mobile devices like on desktop computers, their technical and ergonomical constraints bring new issues to consider. Firstly, according to [1], usage of mobile devices is not as exclusive as using a desktop computer at home. Mobile users can be distracted at any time by their context, hence they want an adaptive and flexible way to access the information they are expecting at a time. Secondly, mobile devices are technically limited: battery life, memory capacity, computing power, screen size, input interfaces, etc... Hence,

the browsing experience and the display of contents must be adapted to ensure their usefulness on the mobile device. Thirdly, wireless access to the internet is too expensive for users to afford wasting long and bandwidth-consuming connections as they could at home using unlimited broadband Internet access or TV.

In order to bring the multimedia web to our mobile devices while satisfying these constraints, we need to adapt the retrieval of multimedia content for these specific platforms and their usage. Our approach is to allow users to browse inside the video content without having to transfer it integrally on the device and to personalize the content. This is made possible by video indexing, as long as a browsing interface can be generated from the resulting indices, providing direct links to access its most relevant segments.

In this paper, we propose a web-based interface system relying on a SEO-indexed video library to bring rich and personalized video content efficiently (by focusing on the information that matters for the user) and adaptively (to the platform's constraints) at anytime (on demand) and anywhere (on the move or at home) to the average end-user (with ease of use). As we are aware that new devices and new types of video contents are constantly appearing on the market, this system is adaptive to new devices and extensible to new video domains. Moreover, its modular architecture makes it possible to integrate new components allowing browsing content in an intuitive, precise and enjoyable manner.

This paper is structured as follows: Section 2 describes our previous work and the proposed application of the system; Section 3 specifies the workflow of the expected application; Section 4 outlines the architecture of the proposed interface system; Section 5 explains the design of the dynamic interface generation; Section 6 describes the implementation of our application on the system; Section 7 discusses the success of our approach by evaluating the application; and finally, Section 8 and 9 describe the conclusions and future work.

II. APPLICATION AND PREVIOUS WORK

As an application of such a system, we have proposed navigation on soccer matches with adaptation to user preferences [2]. We have identified use cases which mobile soccer enthusiasts would benefit from. The idea is to browse

soccer matches on the move after having recorded them at home. Users could then use their mobile device (e.g. during their daily bus journey) to browse the highlights of a match in a constrained/noisy environment, browse the latest exciting events concerning one's favorite players, list matches with interesting specificities, etc...

For this application, we need to extract some valuable metadata (structure and semantics) from the soccer videos recorded on TV. Inspiring from MPEG-7, we have proposed a video indexing scheme and developed tools permitting to segment and semi-automatically analyze soccer videos in order to generate these indices from TV-recorded soccer matches [2]. The extraction process is out of the scope of this paper, but we will summarize the SEO indexing scheme.

SEO is a semi-schema and object-relational based indexing scheme allowing a flexible and efficient way of annotating and indexing video content in XML. This work has been focusing on soccer videos essentially but the same paradigm can be used for different domains as well. Moreover, it is based on MPEG-7 concepts and ideas, which have been widely-accepted as the video description standards by video professionals and institutions. A *SEO* index consists of the following components:

- *Segments*: A segment is an individual spatio-temporal range of a video document that contains indexable contents such as whistle, slow motion replay, and close-up on players' face and near goal area.
- *Events*: An event is a special type of video segment that contains a particular theme or topic such as a soccer goal and foul. It usually embeds annotations and possibly linking to objects.
- *Objects*: An object can be a person, a place, or any other material or immaterial entity that is involved with an event.

Thus, any video item (e.g. a soccer match) contains a definition of its belonging segments, events and objects.

In the SEO model, access to content is brought by the use of domain-specific queries (expressed in the XQuery language) that generate hierarchical summaries from the XML metadata. Some of those queries are depending on the user's preferences in order to personalize the results.

III. WORKFLOW OF THE SOCCER APPLICATION

In order to ensure interactivity and intuitivity for end-users, we have designed the system as a website, allowing users to browse the content by jumping from page to page using hyperlinks. Moreover, this paradigm is easily portable to devices that have limited input capabilities (e.g. few keys, stylus) and can be supported on mobile phones using the WAP technology.

The interface system is session-aware, that means that it keeps the context for each connected user and adapts the

content according to his/her preferences. As shown on Figure 1, connecting on the system leads to a login page. Once authenticated, the user is brought to the homepage which proposes links to queries and video segments for registered domains and latest matches, as shown on Figure 2. For each match, a keyframe and some metadata (e.g. place and date of the match) are displayed. At all times, the user can go back to this page or to the User Preferences management pages shown on Figure 3. These pages allow the user to select his/her favorite players, event and segment types for each sport, using 2-list interfaces.

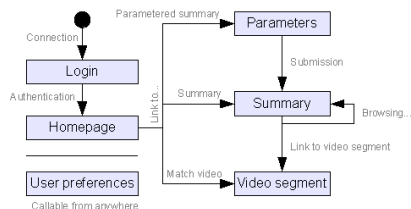


Figure 1: Workflow of the soccer application



Figure 2: Homepage on PC



Figure 3: User preferences management pages

Two types of queries are proposed on the main page: (i) domain queries return summaries which scope a given domain only (e.g. soccer), whereas (ii) media queries return summaries which scope a given video (e.g. a match). Some queries are parameterized, making it possible for the user to customize the results. As seen on Figure 4, when the user selects such one, he/she will be invited to fill in a form of

parameters (a) before displaying the resulting summary (b). This form is already populated in order to propose predefined (and existing) options to the user instead of letting him/her type the corresponding information. The summary is an interactive page containing a hierarchical browsing pane at the top and a details pane at the bottom showing data about the currently selected item. Depending on the type of item, links to video segments or to other summaries can be proposed.

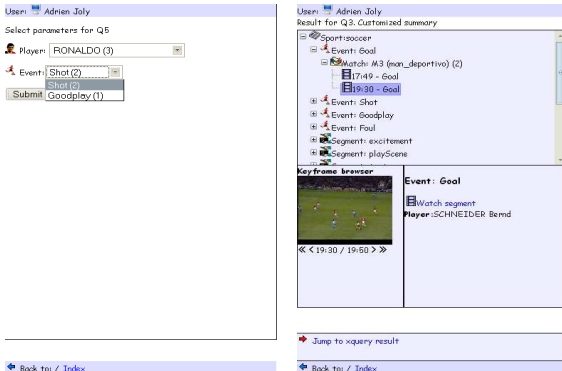


Figure 4: (a) Parameters form and (b) summary

More details about this application and its user evaluation are given in Sections VI and VII. We are now describing the architecture and design of the interface system on which relies the soccer application.

IV. SYSTEM ARCHITECTURE

In order to easily support many kinds of client devices by generating and delivering adapted user interfaces to each of them, we decided to adopt a web-based architecture that consists in “light” clients using a web browser to interact with an applicative server called the *Retrieval Server*. Figure 5 depicts the architecture of the proposed system.

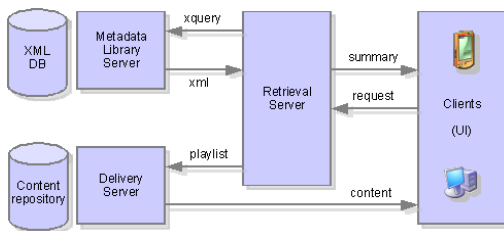


Figure 5: Architecture of the system

Because the metadata library is heavily solicited by this system, access to the metadata has to be handled by a XML database server that ensures reliable and efficient retrieval using XQueries [3]. For that purpose, we have chosen to use eXist [4], a popular and robust open source solution.

The actual delivery of video content must be handled by one or several streaming server(s), according to the type and format of content and the final application. We chose not to focus on streaming issues, hence the proposed system was designed to support any streaming server by extension.

A. Design of the retrieval server

Our system is expected to be extensible to new domains, browsing schemes, platforms and delivery servers. In order to satisfy these specifications, the *Retrieval Server* was designed with the layers listed on Figure 6: (i) the *data access* layer contains the queries that feed the system with data from the metadata library, (ii) the *representation* layer defines the templates that transform raw metadata into their human-readable representation, (iii) the *user interface* layer proposes components that implement platform-specific browsing schemes, (iv) the *services* layer handles the calls for delivery of summaries and content to the user that will be actually processed by (v) the *servers* layer.

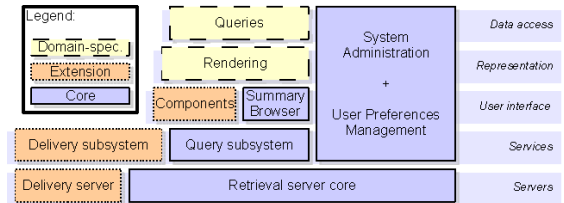


Figure 6: Extensibility of the retrieval server

As depicted on Figure 6, the layers (i) and (ii) can be extended with domain-specific queries and rendering templates. New browsing schemes and platforms can be supported by extending the layer (iii) with components. And content delivery servers can be plugged by adding a handler to the layer (iv) services.

B. Implementation of the retrieval server

The *Retrieval Server Core* located on the layer (v) is a J2EE application based on Java servlets which handle general web logic (e.g. sessions). On the layer (iv), the *Query Subsystem* consist in servlets implementing the GUI (Graphical User Interface) generation engine which drives the data flow from the metadatabase to the final user interface. This will be explained further in Section V. On the same layer, the *Delivery Subsystem* can be extended by servlets to give access to the content from specific delivery servers. In the current implementation of the system, we use the HTTP server as a *Delivery Server*, providing direct download access to content. Hence we implemented two servlets: the first returns URLs to keyframes which are identified by the video identifier and the timestamp, and the second returns URLs to video segments. Both layers (iii) and (ii) rely on XSL transformations. This will be explained further in Section V. At last, the layer (i) consists in XQuery files.

V. DYNAMIC GUI GENERATION PIPELINE

In order to generate summaries adaptively to an extensible pool of queries, domains, browsing schemes and client platforms, the GUI must be dynamically generated. As our queries return XML data that must be processed to generate platform-adapted HTML summaries, we chose to follow the “XML/XSLT pipeline” approach presented in [5]. This

approach consists in processing the XML input data with different XSLT transformations in order to obtain platform-adapted documents at the end of the pipeline. Furthermore, because the numerous extensible layers result in many different chain combinations, the pipeline has to be generated dynamically. In order to achieve this, the GUI subsystem generates the pipeline by building a graph in which transformations are connected according to their specified input and output formats.

Browsing Schemes are defined as GUI components that transform high-level results of queries into platform-adapted interactive summaries allowing the user to browse the contents. The system natively includes a generic GUI component called the *Summary Browser*, which consists of a hierarchical browser composed of a *tree pane* and a *content pane*, as seen previously on Figure 4. It is generic since this tree can match the structure of any XML output from the queries. In this browsing scheme, users can select nodes in the tree of results to display their corresponding metadata and content (e.g. details, keyframes and other related data). For example, if a query returns a set of soccer matches, clicking on a match will show the location and date of that match, some keyframes, hyperlinks to match-specific queries and to the whole match video.

Note that a “*Keyframe Browser*” is natively included in the system as another GUI component. It enables the user to browse the keyframes of a segment.

A. Case of the Summary Browser

The generation of a summary using the *Summary Browser* component is a particular scenario of the pipeline. As it is the only browsing scheme natively included with the system, we will now describe the process of generating a summary from the results of a query.

As depicted on Figure 7, this process consists in 3 transformations:

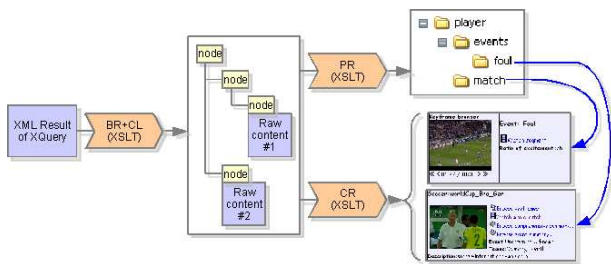


Figure 7: Data flow from the XML query result to the final HTML page using the Summary Browser

1. **Browsing document Rendering (BR) with Content Location (CL):** This query-specific XSL transformation builds up the structure of the summary that will be shown on the final HTML page for browsing from the results of the query. The output of this transformation respects the input format of the *Summary Browser* component; it is a high-level XML format with no platform-specificities.

Moreover, this transformation calls a generic “content location” transformation that will embed the content (e.g. reference to keyframes and textual details) associated with nodes of the tree structure, in order to leave their rendering for later. This transformation permits to separate the structure from the content while keeping linkage information using identifiers for late binding.

2. **Page Rendering (PR):** This transformation renders the final platform-adapted DHTML page from the high-level tree structure resulting of the previous transformation. This page contains a *treeview* component and the scripts driving the browsing logic for the final summary. This transformation is actually the instantiation of the *Summary Browser* component.
3. **Content Rendering (CR):** This transformation is a second pass on the output of the BR+CL transformation. It extracts the embedded content elements generated by the *Content Locator* and delegates their rendering to domain-specific templates defining the representation of those elements in the expected output format (HTML).

Then, the rendered content is integrated to the page (by merging them) to obtain the final platform-adapted summary that will be returned to the client. The DHTML code of the page drives the browsing interactivity, content being bound to their corresponding tree nodes thanks to the identification applied by the CL transformation.

The main strengths of this design are that: firstly, we abstract the platform specificities in the first transformation, as it only defines the way to structure the results of a given query in a high-level format. Thus, it is easy to add support for new client platforms or change the GUI layout by implementing different versions of the *Page* and *Content Rendering* transformations only. Secondly, the rendering of content is delegated to domain-specific templates. Hence, it's easy to add support for new domains; the main transformation engine remains unchanged. Thirdly, the separation of the tree and the linked content makes it possible to implement a “PULL” version of the *Summary Browser* that would download content on-demand from the server instead of downloading everything (“PUSH” of the contents from the server).

B. System configuration model

This section will describe the configuration model which the extensible GUI generation pipeline relies on. This model defines the entities that the system deals with in order to generate adapted interfaces dynamically. It is stored as a XML document in the database. Extending the system consists in merging extensions in this common system configuration XML tree.

As depicted on Figure 8, in the *system* configuration tree are defined *domains* and *platforms*. For both of them, instances are hierarchically linked using a reference to the *parent* identifier. In the domain hierarchy, each *domain* node inherits the *queries* that are defined for its ancestors. In the *platform*

hierarchy, each *platform* node inherits the *transformations* that are defined for its ancestors.

A *domain* is defined by:

- *Queries* are proposed on the main page to provide summaries that scope on the domain,
- *Media-queries* are defined like *queries* but are proposed for each video items of the corresponding domain (e.g. the “event summary” and “comprehensive summary”) and,
- Its *renderer*: a XSL transformation that renders the metadata and associated content from the SEO components (segments, events and objects) that are defined for this domain and returned by the queries.

A *Query* is defined by:

- Its *presentation* consisting in XSL transformations that will convert the XML result of the query to the input format of the rendering component that will be used to generate the final interactive summary. Note that different transformations can be proposed, depending on the *target* platform.
- An optional *form* consisting in a *query* and its *presentation* (defined as above), for parameterized queries only. The query will ensure the retrieval of data that will be used to populate the form generated by a XSL transformation.

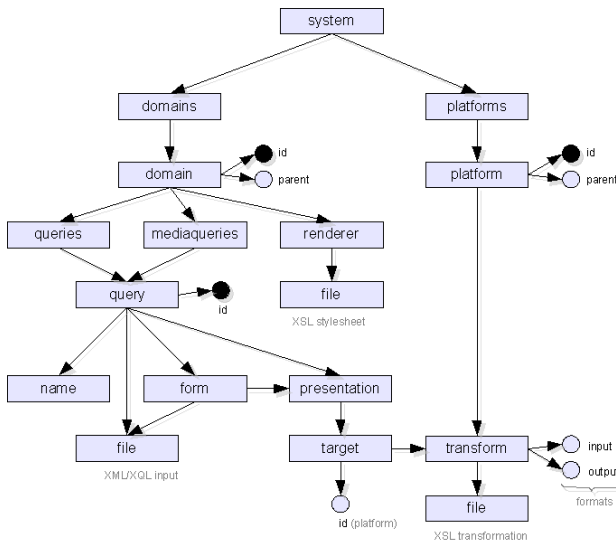


Figure 8: System configuration scheme

Platforms are defined as a set of platform-specific XSL transformations for given input and output *formats*. Each transformation is actually the implementation of a GUI component (like the *Summary Browser*) that will generate the final user interface from formatted query results. The platform hierarchy enables the GUI generation engine to elect higher-level (and thus, less precisely adapted) platforms for the case where a perfect match between a required *target*

platform and an existing platform definition could not be found. As an example, we can consider the following platform hierarchy: *pfm.pc*, *pfm.ppc*, *pfm.ppc.vga*, etc, where *pfm* denotes platform; *pc* corresponds to Personal Computers (desktop), *ppc* stands for Pocket PC and *vga* is a special kind of *ppc* with a high-definition screen. If we are using a Pocket PC client with VGA screen, but that no version of the *Summary Browser* has been implemented specifically for VGA screens, the standard Pocket PC version will be used instead.

The *Format* identifiers are used to match inputs and outputs in order to build transformation graphs automatically. Like platforms, formats are also identified hierarchically, where the descendants specialize their ancestors, giving more flexibility for the GUI generation process. Note that, contrary to *platforms*, *formats* are not explicitly declared. They are implicitly defined by their hierarchical identifier. As an example, we can consider the following format hierarchy: *fnt.browser*, *fnt.html.basic*, *fnt.html.v4*, where *fnt* denotes format, *browser* denotes the “*Summary Browser*” input format, and *html* format could be *basic* or advanced (*v4*) depending on the target web browser.

A *transformation* file is specified given its *input* and *output* formats. During use, the system will build the transformation pipeline required to render the GUI as a chain of transformations in which the inputs are optimally corresponding to the outputs for the given platform.

VI. IMPLEMENTATION WITH SOCCER VIDEOS

The system has been implemented and deployed on Tomcat as a web application. As seen on Figure 9, the *Summary Browser* GUI component has been implemented in DHTML (HTML + JavaScript) for both modern browsers on PC and Pocket Internet Explorer on Pocket PC.

In order to evaluate the system, we have implemented the “soccer” domain (queries and templates) and added 3 SEO-indexed soccer matches in the video library. This application supports 6 summaries, including 2 which are parameterized, and 1 which is based on user preferences. We will now describe the summaries Q1, Q2, Q3, Q4, Q5 and Q6.

Q1. Comprehensive Summary: This summary lists the « play-break tracks » of a given match. Each track consists in a “play” and a “break” segment, the “play” phase being interrupted by an event (e.g. foul, goal...). A “play” segment describes the cause of an event whereas a “break” segment describes its outcome.

Q2. Events Summary: This summary chronologically lists the events happening in a given match.

Q3. Players by Team: This summary lists all video segments in which a player appears as part of one of his teams, for every player of any team.

Q4. Player Summary: After selecting a player, this query returns personal and strategic details (e.g. position) and event

segments related to this player. Segments are grouped by event type and match.

Q5. Exciting Matches: This summary provides a list of matches filtered by their custom number of goals and fouls.

Q6. Personalized Summary: Basing on the user's preferences, this summary shows the latest video segments related to his/her favorite players, types of events and segments.



Figure 9: Web-based Video Retrieval System Accessible for Desktop and Mobile Devices

VII. USER EVALUATION

To prove the effectiveness of our proposed interface system, we have conducted a user evaluation on the soccer application relying on this system with a group of 53 university students. The aim of this survey is to gather users' feedback on the effectiveness, intuitiveness and enjoy-ability of the system for this application. Each criterion is measured uniformly by quantified "strongly agree, agree, disagree, and strongly disagree".

As shown on Figure 10, an average of 90% of the surveyed users agreed with the effectiveness, intuitiveness and enjoy-ability of the system for the proposed soccer application, and about 15% of them strongly agreed. The most appreciated summaries were Q2 (Event Summary) and Q6 (Personalized Summary).

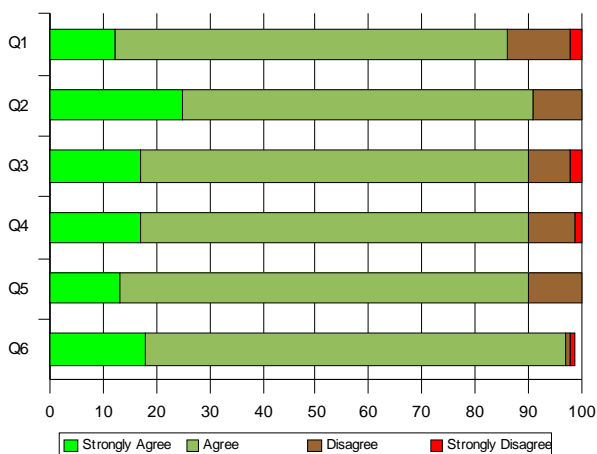


Figure 10: User acceptability of summaries

VIII. CONCLUSION

In this paper, we have proposed the architecture and design of an extensible video retrieval and browsing system. The architecture is intended for use with a XML-based video content indexing scheme, and we have shown its effectiveness with SEO-indexed soccer videos. The design is highly modular, allowing adaptation to various domains, browsing schemes and any web-enabled platform/device.

The major contributions proposed in this paper are: (i) the extensible multi-layered architecture of the system, (ii) the *Summary Browser* as a hierarchical browsing scheme allowing to browse the results of virtually any query in a rich and user-friendly manner, (iii) the graph-based GUI generation pipeline, and (iv) the system configuration scheme including domain description and platform adaptation.

Our approach expands the hyper-navigation paradigm that is used on today's websites (pages made of text and images) to video content in order to browse it in a non linear manner. Moreover, our evaluated implementation demonstrated that such an approach is realizable and effective.

This system could be the platform for new services brought to end-users for video hyper-navigation and summarization. It is suitable for many specific applications that could bring profit to content providers by selling more content and/or by proposing highly-targeted advertising to their clients.

IX. FUTURE DIRECTIONS

As the architecture and design of the system proposed in this paper has already shown its robustness in the scope of its application in a soccer video library, we propose some future directions to improve it further: (i) add new browsing schemes (e.g. using thumbnails or chronological representation), (ii) add new data rendering tools (e.g. charts), (iii) consider user's location and environment for delivery of targeted and adapted content, (iv) add user community and exchange features (e.g. forums, and tagging), and (v) support custom query creation.

REFERENCES

- [1] H. Knoche and J. McCarthy, "Design requirements for mobile TV," *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pp. 69-76, 2005.
- [2] D. Tjondronegoro, *Content-based Video Indexing for Sports Applications using Intergrated MultiModal Approach*. Melbourne: Deakin University, 2005.
- [3] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon, "XQuery 1.0: An XML Query Language," *W3C Candidate Recommendation*, 2006
- [4] W. Meier, "eXist: An Open Source Native XML Database," *Web, Web-Services, and Database Systems. NODe*, pp. 7-10, 2002
- [5] M. Butler, *Current Technologies for Device Independence*.: Hewlett-Packard Laboratories, 2001.