Active Server for the Management of Structured Documents Link Integrity

Abraham Alvarez[†], Youssef Amghar, Richard Chbeir.
INSA de Lyon – LISI, 7 avenue Jean Capelle.
69621, Villeurbanne – France.
Phone: +33 4 72 43 85 88, Fax: +33 4 72 43 87 13
{aalvarez,rchbeir}@lisi.insa-lyon.fr,amghar@if.insa-lyon.fr

Abstract: In the electronic age, World Wide Web, IntraNet and Webware are becoming key concepts in the way we read, write, and deliver information services. In most cases some enterprises are taking advantage of them by using Corporate Web-based Systems, to scatter and to share structured documentation. Nevertheless, preserving the link integrity of a document repository is one of crucial problems. Precisely, whenever documents' links are modified, the potential risk to destroy the record of a linked text occurs. An unwanted problem is the "Error: 404 file not found". In this paper, we focus on this issue and we try to remedy this shortcoming. We believe that by enhancing the Web Server capabilities (e.g., monitor of events, processor of events and trigger components), we can provide an active mechanism for the link integrity management. This study compared with existed approaches proposes a solution based on active functionalities.

Key words: Active Web Server, Referential Integrity, Triggers, Web-based Systems, Active Database Manager System.

[†] This research paper is financially supported by The National Council for Sciences and Technology of MEXICO (CONACYT).

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35614-3_21
C. Rolland et al. (eds.), *Engineering Information Systems in the Internet Context*

1. INTRODUCTION

The business contributions of Intranet applications have proved that this technology is a golden approach to support Corporate Web-based Systems as stated in [1,2]. Increasing their number, all kinds of organizations are taking advantage of Intranet to propagate structured documentation (e.g. technical, medical, academic, etc.). Intranet technology is as well used to enhance organizations' business strategies [3]. The current Web and Intranet services are short of referential integrity support. Observable examples are broken-links and dangling-references, arguably one of the most unwanted problems in distributed domains.

Another frequent problem is resource migration. The experience has shown that the Web resources often "migrate" during the re-organization of hardware resources. e.g., as a result of an increasing in the volume of information published by a given site.

The question is how to preserve the link integrity, at any time, document links can be altered some way, occurring the potential risk to destroy the record of a linked text (broken links). A typical example is the unwanted "Error: 404 file not found". In this paper, we focus on the management and the automatic reparation of documents' coherence across an Intranet framework. This study proposes a solution based on active functionalities. It proposes the definition of an Active Web-Server of Documents. Our approach is based both on the Active Database (ECA rules paradigm) and Java Technology; the rules are used as alert mechanisms. The rest of this paper is organized as follows. The motivation of this study is argued in section 2. Different approaches for managing link integrity are exposed in section 3. Section 4 introduces our approach for turning a classical Web Server to an Active Web-Server, which functionalities are explained. A document model for managing links is considered around of a case-study. The ECA rules definition and implementation architecture are covered. Finally, a conclusion and future perspectives are discussed.

2. MOTIVATION

One of the causes of broken links is the volatility of the medium. thousands of manipulations are done within very short time. Classic manipulations include:

- **Deleting pages:** the action to delete pages containing links has a true impact over the document database. The referenced documents to the document deleted will not be longer valid.
- **Splitting pages:** means dividing a page into two or more pages. Since it is very difficult to know which one of the split pages we have to point to.
- **Renaming a file:** occurs when the contents of a page are broadened or narrowed. File-name renaming is not equivalent to moving a file because all the referencing links are still correct (only for embedded links), however each of the anchor names of these links may now be incorrect.

Knowing the factors of risk, the user claims a reactive work environment including a notification mechanism like a "pop-up message" when an edition operation is done. Edition operations include: changes in the link address, insertion, and deletion, of text nodes. Finally, we could argue that: the existing solutions called "link managers" lack the active capabilities.

The main motivation of this study is to provide a dynamic mechanism based on active functionalities to warrant the link integrity. It mechanism includes both the definition of an Active Server of Documents, and the support of Active Database technology.

3. SOLUTIONS AND LIMITATIONS

In the literature, we have found several research projects addressing referential link integrity problem. Two main categories of solutions have been proposed [4,5]:

3.1 Preventive solutions

- **Forbidding changes:** is a preventive strategy, the modification of documents is not authorized [6]. Links pointing to documents or

part of documents should not fail. This approach does not provide a protection in cases where the host information is changed.

- **Versioning:** is one of the main issues in hypermedia field adopted by Vitali [7] and Nelson [8]. Therefore, versioning philosophy does not participate in the application "by default".
- **Embedded links:** embedded links avoid links errors in two ways: the first way embeds the source end point of a link into a document, so that a wrong reference cannot occur. The second way is embedding of name tags, e.g. <A name="xxx"> tags supported by HTML, to help with the referential integrity of destination end points. Davis [9] proposes this approach.
- **External Links:** Generally, they are stored in external link database, where each link is a record entry into a link table. The idea of this approach has been exploited in hypermedia systems [10,11,12]. A similar approach is described in [13] where the authors consider a separation between the document content, links, and their content. The weighty maintaining of external link bases is a disadvantage, even a link resource changes, and external links risk pointing to the wrong reference place.
- **Link attributes:** links must be to describe the target node and provide some information concerning their location, nevertheless some semantic characteristics are not expressed. SEPIA is hypermedia system that provides semantic types [14]. Oinas-Kukkonen in [15] proposes to integrate link attributes like semantic link types and link keywords. Link attributes properties associated to links provide knowledge about interrelationships between pieces of information. Attributes also provide the user a way to know or preview the target before activating the link, [16]. The main weakness of these approaches is the volatility of the medium.

3.2 Corrective solutions

- **Forward mechanism:** is a method that informs readers at any time when a linked resource has been corrupted or changed, and some times will automatically point the reader to the new location.
- **Universal Resource Names or URNs:** are persistent identifiers for information resources. An example is a formal proposal being

developed by the Internet Engineering Task Force (IETF), where the aliases are called URNs [17].

- **Relative references:** HieNet [18] is a link mechanism conceived to generate new links based on previously created. This approach is inspired in Salton's Vector space model [19] and Bernstein work [20].
- **The paradigm of Agents:** In the WWW domain an agent must examine the hug of html documents contained over Internet in order to locate the dangling links [21]. While, Leslie [22] introduces a Distributed Link Service (DLS), the agent helps to users for browsing and information discovery activities.

4. ACTIVE WEB SERVER DEFINITION

The interest of building an Active Server of Documents is to add an active behavior in order to allow the treatment of specific events as the documents operations (moving, renaming, content-modifying, deleting, etc). The aim is, to reduce the occurrences of unexpected changes and to repair some incoherencies. An active server must be able to:

1. detect events related to operations on documents occurred on clients;
2. verify conditions on global consistency querying document base,
3. correct integrity violations, maintain coherence, propagate updates and notify users using triggers.

4.1 Active Model for Documents Web Server

An active relational or object database management system consists of adding trigger mechanisms [23,24,25]. Appeared in the seventies, the notion of trigger has been generalized with the notion of active rule based on the ECA formalism. The semantics of an ECA rule is as follows: when an event E is produced, if the condition C is satisfied, then the action A is executed. The rules allow database designers to specify the active behavior of the database application that provides the enforcement of database integrity. Concerning this study, the rules

are considered as a means to translate the policies of document applications. The components of a rule pattern are defined below.

4.1.1 Event

An event is an occurrence in the database or application's environment. In this way, "the detection of changes within the application's scope is performed using events." e.g. when a document is altered, the documents server reacts to this change, which is materialized by with an event through the network. A type of event describes the situations that can be recognized by the system.

1. **Primitive.** these belong to one of the following classical database operations: insert, delete, update. For document database, the set of operations is extended by the replace operation. These operations are generally described as follows:
   - **Insertion:** every time new resources (XML or HTML documents) are inserted, new target anchors are created containing a reference to the appropriate resource description.
   - **Update:** updating documents or link documents can lead to a global inconsistency's document database on the server.
   - **Suppression:** before the specified web based resources can actually be deleted, their entries in the resource description entity with the appropriate URLs have to be deleted.

For our purpose, primitive operations, which can be done on documents leading to potential inconsistency's document database:

- **Move or rename:** moving a page involves physically moving the page in question, and keeping consistent in the web, links that refer to this page. It is possible to automatically update links that point to a moved page as long as the above restrictions or are honoured when links can be automatically repaired.
- **Modification:** documents and links modifications can occur either at the whole-document level or at internal level. Whole-document changes include the moving or renaming of documents or host machine, name or domain name changes. Internal changes include moving, editing, adding text, etc.
- **Deletion:** documents and links deletions also occur at the whole-document level or at the internal level. Whole-document deletions include the usual deletion. Internal deletions in documents can include the removal of fragments of text or other data.

- **Link Missing:** attempt to find the missing resource, such as href, ftp, http, telnet, etc. they are typically recognized by a "mark up" code, i.e. "Tag" already embedded in the text.

2. **Temporal.** The corresponding event represents information referencing the time. These references can be absolute (on January 15 of 2002), relative i.e. between an interval of time (± 15 minutes, between 00:00hrs and 00:20hrs) and periodical (once a day, weekly, monthly, etc).

4.1.2 Condition.

A condition is a predicate or a query over a database (such as a comparison between object attributes). The condition consists of controlling the action execution. In the context of documentation coherence, conditions may concern link verifications. When document database are structured through XML, a condition may result from Xpath query (condition on document database) or a SQL query (condition on consistency).

```
Define rule Insertion_Link
On insert Document.Element
if  (new.element = true)
   then add Document.Element
      where doc_status = valid
```

4.1.3 Action.

An action can produce one or more events. Corrective solution tries to solve situations when we move or delete a document. We have three ways of behaving responsibly:

1. we can leave some form of forward reference to the new document,
2. we can inform anything that points at us about the change,
3. we can update some name server through which our document is accessed.

To summarize the three components of a rule on the documents' active server context, we present the rule semantic parts: **events, conditions and actions** that can be predefined from Table 1 to Table 3.

Table 1 shows the different predefined actions to perform whenever the creation of links in documents and the insertion (storing) of a document in the database event occurs.

Table 1. Predefined rules for operations of link creation and storing document.

Event	Condition	Action
Insert internal link	valid link source	ensure the target point being valid i.e. a document already exists. If it is valid, insert the new link
	not valid link source	tag the link as disable.
		send a message to inform the user that link is disabling until being valid
	<A name> exist	search the tagged links stored in the history that have been requested to refer this link before <insertion>, finally add the link.
Insert external link	valid link source	"manual verification", launcher the query to other sites (msn, google, etc) and propose a link by similarity or a precedent link
	not valid link source	tag the link as disable
Insert document		find the instances on tagged links that have been trying to point to this document.
		enable the tagged links and do the insertion.
		send a message to inform that link is now valid

Table 2 shows the different predefined actions to perform whenever a user achieves editing operations. They mainly concern operations that modify a document by updating links or fragments of document. The fragment updating may impact as well on document structure as content portion (image, text, video,) leading the possibility of breaking a link of another document to the updated document.

Table 2. Predefined rules for operations of updating of links and document.

Event	condition	Action
Edit internal link	valid link source	verify the target point is still valid. If it is valid save information in the Log journal.
	not valid link source	tag the link as disable and create a new fragment version and send a message to the own user to inform that it is disable until being valid.
	<A name> exist	verification of all referring links to this resource being valid.
Edit external link	valid link source	"manual verification" by query of the target point is no longer valid after <edition>.
	not valid link source	tag the link as disable and create a new fragment version and send a message to owner to inform is disabling until being valid.
Rename document		create a new instance and ~alias in the database

Change path-name	Tag the root path-name and create an ~alias
Move document location	create a new instance and alias in the database
Modify content	create a new fragment version of modified fragment.

Table 3 shows the different predefined actions to perform whenever a user achieves deleting operations. An operation of suppression can have some critical consequences on global coherence of the document database. The fragment suppression may impact all links pointing on it. Policies must be defined and coded by rules.

Table 3. Predefined rules for operations of links' and documents' suppression.

Event	condition	Action
internal link suppression	valid link source	tag the link as disable (put a flag).
	not valid link source	create a version of the link
	<A name> exist	look in the whole document the possible links pointing to this link.
		create a version of the link.
		send a message corresponding to the suppression
external link suppression	valid link source	tag the link and create a version of the link.
Suppression of document	none	find all references pointing to this document.
		tag the links as disable (put a flag).
		create a new document version.

In the framework of document manipulation, an operation performed on a client of the network will raise an event, such as suppression of internal link, for example. This event will trigger the following rule: *on suppression of the internal link if the source of the link is valid and then tag the link as disable;* this rule is activated at the Active Documents Server level. The condition is verified with a query through the document database. The action is achieved on the server and consists on the update link database. The query can be expressed with SQL as well as Xpath, XQuery [26], XQL [27].

4.2 Data model for document and link representation

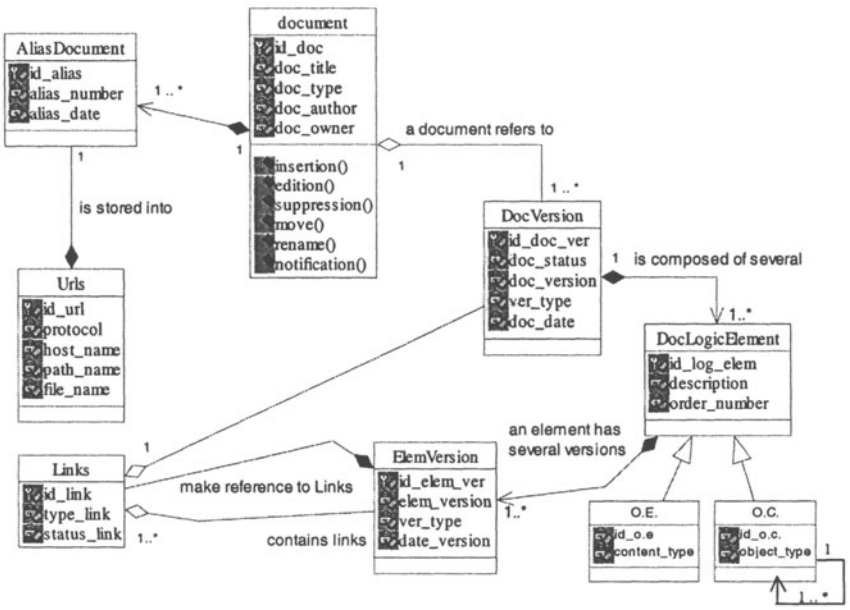

Figure 1. Document Model.

As first step, it is necessary to create a data model to support the links manipulation, e.g. edition, suppression, move and rename operations. In particular, links must represent a foundation of the data model as well versioning. The basis of our model is that: A Document {Document Class} is composed of versions {DocVersion Class}. This class is composed as well of several Logics Element of Document {DocLogicElement Class}. There are two types of "objects" elementary objects (EO) and complex objects (CO). An elementary object can be a {title, title section, paragraph, section, sub-section, etc}, and a complex object is an EO composed by other elementary objects. A logic element {DocLogicElement Class} has several versions {ElemVersion Class} each element of this class contains links {Links Class} either make reference. Finally a document {Document Class} is composed of Alias stored into URLs.

4.3 Example

In the industrial context, a Document Web-Based System related to the technical documentation is a mean to connect the life cycle product with the project planning areas: such as designing, manufacturing, maintaining and assembling of a manufactured product.

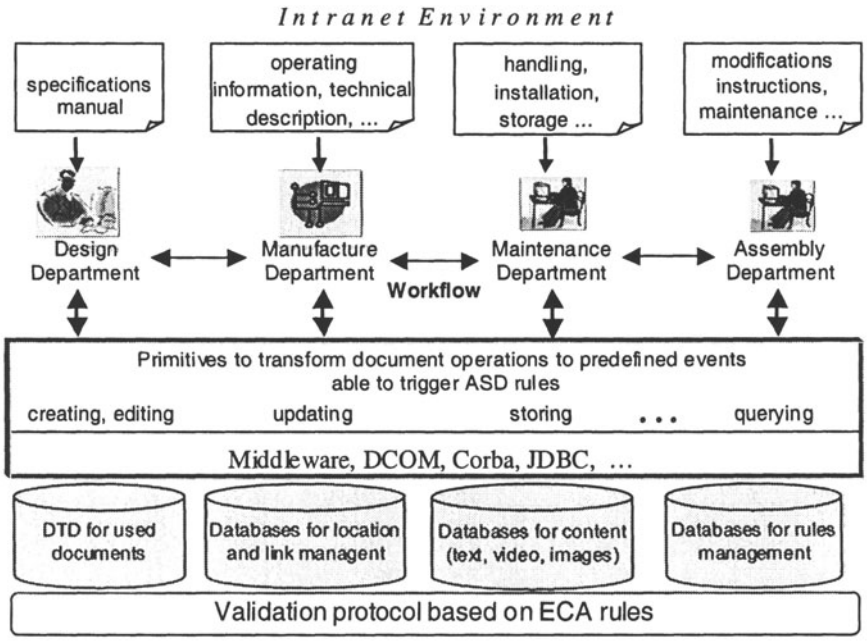

Figure 2. Interrelation of the major phases of product manufacturing

The set of relevant information for a particular task is produced during the life cycle of the product, e.g. manual specifications for a design department, modification instructions for maintenance department and so on. All documents are interrelated by links over all process and any changes can impact notably to other department activities. This process is depicted in the Figure 2.

Let us consider an European manufacturing enterprise of Video Cassette Recorders. This enterprise produces only for the (EC) European Community basing its manufactured products on European electric characteristics i.e. 220V & 50Hz.

A new expansion project aims to buy recorders to the North American area. Considering this requirement the electrical characteristics must be modified as American electric characteristics are 110V and 60Hz. That leads to change all document parts (fragments) containing data about these characteristics. Generally, all documentation concerning life cycle of manufactured products must be revisited (may be some new links must be created). For example, a new annex is added to complete the instruction manual. This annex contains the new product specifications concerning the electric transformation involved by the North American sector. Taking this change into consideration, the new technical annex has also to be created by a new "reference link". The original annex makes reference to a new technical annex. When the user modifies the original annex, the validation mechanism has to be executed by the system automatically creating the new reference link. A rule has been executed to create and validate the new link.

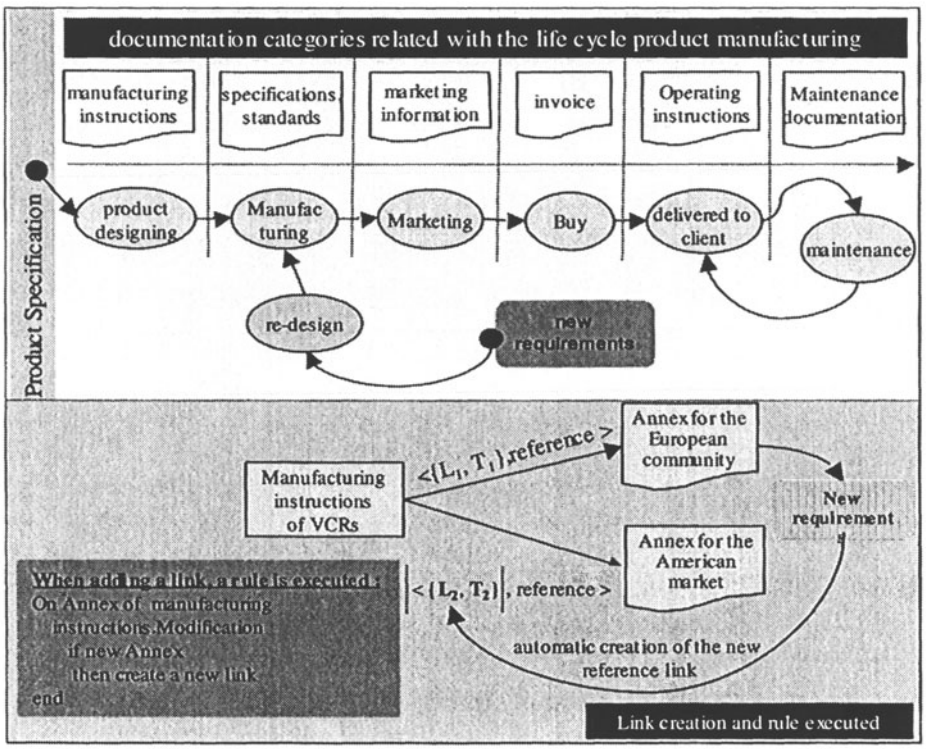

Figure 3. document generation in product life cycle and rules execution.

4.4 Implementation issues

Rather than to define a new Web Server Architecture, we would like to enhance the capabilities of a Classic Web Server, by the integration of an Active Module. The components of an Active Module are represented in the Figure 4.

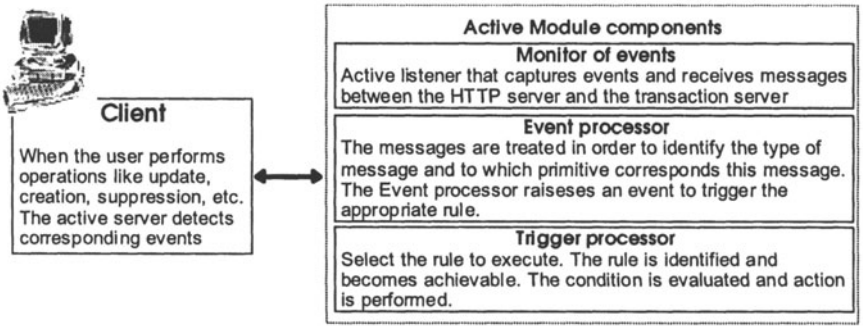

Figure 4. Active Module Components.

Monitor of Events: it works as an active listener (daemon process) capturing events and messages between the HTTP server and the transaction server. Monitor functions are basically: listening and classifying events. An event can occurs when a user requests for a transaction to Web Server. The requests can be simple (e.g., displaying a document, requesting a URL address, retrieving a local file, etc.) or complex (e.g., executing a script, an applet, accessing data via SQL, launching a trigger, etc.). Once the event is captured, it is classified in order to be treated by the processor of events.

The Processor of Events: This module, which treats and identifies the incoming events from the Monitor of Events, in order to processes the allocated parameters of the rules associated with the events. Normally an event e is generated when an operation is requested. An event e specifies when a rule should be triggered, the Condition c is a query that is evaluated when the Event e occurs, this is the namely ECA formalism (Event-Condition-Action). Furthermore, the operations can be associated with a rule. Once the rule is associated to the event, a message is submitted to next module to be triggered depending the event type. The task of this module finishes until the parameters were transferred to the Trigger Processor.

Trigger Processor: It is responsible of the execution of a task when a condition is accomplished. The rule execution carries out when the trigger mechanism launch an action. The commands like: edition, update, insertion are considered as actions, e.g., to verify that all references appointing to a document section are valid and vice-versa. The rules, data, and the links are stored in separated databases. After triggering the rule, the active module process is ended and passes to the client.

Finally, the client is the front part in the client/server architecture. In this scenario, the user is the interactive part between the server and its components. He can request some manipulations as: querying, updating, creating, deleting and other documents or fragments modifications.

Until this point, we have only described the components of the Active Module. Now, in the next section we will to describe the rest of the architecture components.

4.5 Architecture

Concerning to the integration and Web-database access, a grand number of methods exist, CGI scripts, actives pages, and database call http. Additionally, Java Applets using JDBC, Java Sockets, Java Servlets, Remote Method Invocation, and CORBA. This variety of methods enhances the Web Server capabilities giving a dynamic functionality: to create database connections, to execute queries and transactions, etc.

A basic architecture configuration of a classic Web server is depicted in the figure below.

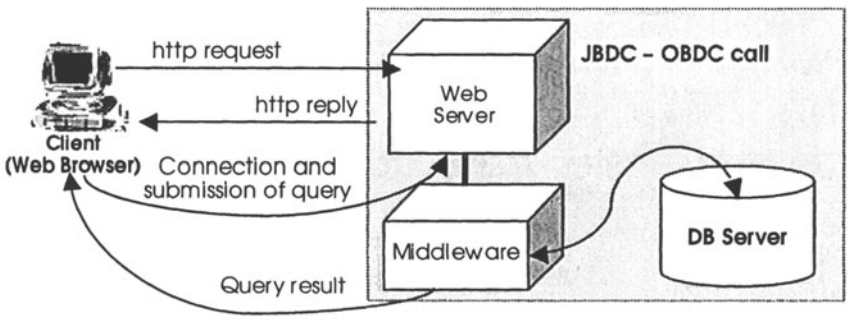

Figure 5. The basic configuration for a classic Web Server.

Client or Web-browser this function as clients, asking Web Servers for information by using HTTP protocol. Web Browser or clients are increasingly being used to support internally developed Corporate Web-based Systems and Applications. Some common tasks that can support the Web Browsers includes:

a) Viewing documents created on heterogeneous platforms.
b) Creating and editing documents
c) Delivering documents

The HTTP Server (also called Web Server). Web Servers store, maintain and deliver information to client via HTPP protocol. The HTTP-server replays to user request through the client, sometimes returning a html page, a plain text file, a result of search, etc.

HTTP protocol. HTTP is the most popular protocol used for access and retrieval of web pages. The Web-browser, asks to the Web server to retrieve some information via "get request" command. The information exchanged by HTTP can be any type of data (text, video, images, sound clip, etc). In the database connectivity domain, different approaches have been presented. These approaches include Java applets, Java sockets, Servlets Remote Method Invocation, CORBA and mobile agents technology. Stavros presents a comparison in this domain evaluating the performance and programmability aspects [28].

We are not interested to define a new Web server architecture. Rather, we would like to provide some elements for enhancing the web server capabilities. i.e. by turning a Classical Web Server to an Active Web Server. Under this framework we have adopted an architecture based

on a hybrid model: an Active Database Manager System and an Active Module.

Herein, two cases are presented, the first case refers to the typical operations as: INSERT, UPDATE and REMOVAL or RELOCATE a link), they are treated directly by the Database Manager System. The second case corresponds to namely "document events", these events correspond to special manipulations not supported by a DBMS e.g., suppression, insertion, edition of internal-external links, and other elements of a document; these events are supported by the set of ECA rules already described in the section 4.1.3 as from tables 1 to 3. These operations normally are supported by the set of ECA rules and executed by triggers mechanism that interact with the active module that will control the integrity of documents and links. The Active Server Module "listens" the events of middleware and the collection of databases. The data, the links and rules are stored in separated databases.

The role of middleware is to accept client requests, execute them on the database server, and return the results to the client. Middleware can be used in the middle as the Infrastructure to allow applications to exchange information between the communication mechanisms and the platforms. Middleware technology allows the Integration at different functional levels: System, Information, Application and Network Integration. Finally, the general architecture is depicted in Figure 6.

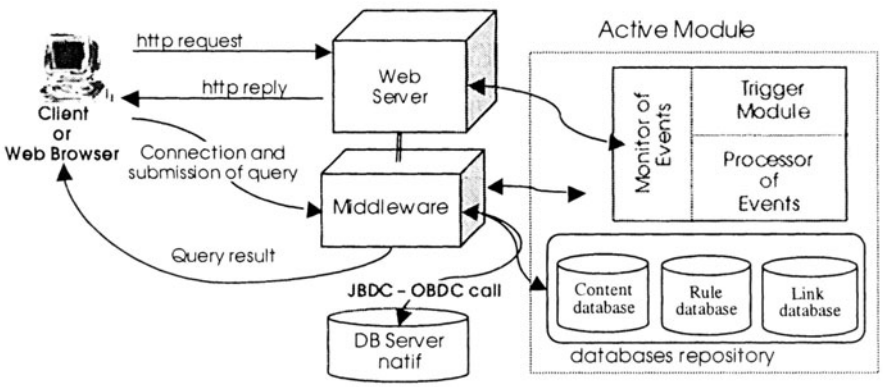

Figure 6. General architecture of Active Server of Documents.

5. CONCLUSION AND DISCUSSION

In this paper, we have tackled an important aspect of Corporate Web-based Systems: the link integrity aspect. We proposed here a robust mechanism able to prevent and repair, in an automatic way, the link integrity between documents. This mechanism is supported by hybrid architecture that includes both features, an Active Module to enhance the Web Server capabilities, (e.g., monitor of events, processor of events and trigger components) and an Active Database Manager System to define and manipulate triggers. We believe that this combination supports the lacks of links managers and Web applications and provides a fine level of granularity under a collaborative framework.

To built a flexible and scalable system, is appreciate to consider the rules maintenance and to allows the addition, suppression or rules modification. The maintenance requires the development of algorithms e.g., validation of the coherences' rules. We consider in the future to explorer this way of research.

6. REFERENCES

1 V. Balasubramanian, Alf Bashian, Daniel Porcher. "A Large-Scale Hypermedia Application using Document Management and Web Technologies", Hypertext 97, April 6-11, 1997

2 V. Balasubramanian, Alf Bashian "Document Management and Web Technologies", Comm of the ACM, vol. 41, No. 7, July 1998

3 Bonifatti Angela, Ceri Stefano, Paraboschi Stefano. "Active rules for XML: A new paradigm for E-services", 1st Workshop on E-services Proceedings, co-held with the 26th The VLDB Journal, pp. 39-47, September 2000

4 Ashman Helen. Rosemary Michelle S. "Computing Survey's Electronic Symposium on Hypertext and Hypermedia:Editorial." ACM Computing Surveys Hypertext and Hypermedia Electronic Symposium", Vol. 31 (4es), Dec 1999.

5 Ashman Helen. "Electronic Document Addressing: Dealing with Change", ACM Computing Surveys, Vol. 32, No. 3, September 2000.

6 Davis Hugh C. "Hypertext Link Integrity", ACM Computing Surveys, Vol. 31, Number 4es, Dec 1999.

7 Vitali Fabio. "Versioning hypermedia", ACM Computing Surveys, Vol. 31, Number 4es, Dec 1999.

8 Nelson Theodor H.. "Xanalogical Structure, Need now more that ever: Parallel Documents, deep links to content, deep versioning, and deep re-use". ACM Computing Surveys, Vol. 31, Number 4es, Dec 1999.

9 Davis Hugh C. "Referential Integrity of Links in Open Hypermedia Systems", Hypertext '98, Proc. of the Ninth ACM Conference on Hypertext and Hypermedia. June 20-24, Pittsburgh, PA, USA, 1998.

10 Andrews, K., Kappe, F., Maurer H.: "Hyper-G, Towards the Next Generations of Network Information Technology". Information Processing and Management. Special issue; Selected Proceedings of the Workshop on Distributed Multimedia Systems. Graz, Austria, 1995.

11 Maurer H. Hyperaware, "The next generation Web Solution", Addison-Wesley, Reading, Massachusetts 1996.

12 Haan Bernard J., Paul Kahn, Victor A. Riley, James H. Coombs, and Norman K. Meyrowitz. "IRIS Hypermedia Services" in Comm. of the ACM, Vol, 35, Isssue 1. pp. 36-51, January 1992.

13 Wilde Eric, Lowe David. "From Content-centered Publishing to a Link-based View of Information Resources", International Conference on System Sciences, Maui, Hawaii, January 2000.

14 Streiz, N. Haake, J. Hannemann, J., Lemke, A. Schuler, W. Schütt, H. and Thüring, M. "SEPIA: a cooperative hypermedia authoring environment", Conference on Hypertext, pp. 11-22, New York, N.Y. USA. 1992.

15 Oinas-Kukkonen Harri, "What is Inside a Link?", Communications of the ACM. pp. 57-66, Vol. 41, No.7. July 1998.

16 Thüring, M., Hannemann, J., and Haake, J.M. "Designing for comprehension: a cognitive approach to hypermedia development", Communications of the ACM. pp. 57-66, Vol. 38, August 1995.

17 Berners-Lee T. "Universal Resources Identifiers in WWW: a unifying syntax for the expression of names and addresses of objects on the network as used in the WWW", pp. 59-94, (1996.

18 Daniel T, Chang, "HieNet:User-Centered Approach for Automatic Link Generation", Hypertext'93, Proc. November 14-18, pp. 256 – 259, Seattle, Washington, USA. November 1993.

19 Salton G, "The smart document retrieval project" in ACM Proceedings of the 4th international SIGIR conference on Research and development in information retrieval, Chicago, Illinois, September 1991.

20 Bernstein Mark, "An Apprentice that discovers Hypertext Links", In Hypertext: concepts, systems and applications. Proceedings of the European Conference on Hypertext. INRIA, Versailles, France; pp. 212 –223, November 1990.

21 I. Varlamis, M. Vazirgiannis. "Web Document Searching Using Enhanced Hyperlink Semantics Based on XML" IEEE, 2001.

22 Carr Leslie A. "Link Services or Links Agents", Hypertext '98, Proc. of the Ninth ACM Conference on Hypertext and Hypermedia: Links, Objects, Time and Space - Structure in Hypermedia Systems, June 20-24, Pittsburgh, PA, USA, 1998.

23 Dayal, U, Buchman A, and McCarthy, D. "Rules are objects too: A knowledge model for an active object oriented database system.". In Proceedings of the Second International Workshop on OODBS. LNCS 334 K. Dittrich, 129-143. 1988.

24 Windom, J. and Finkelstein, S.. "Set oriented production rules in relational database systems", Proceedings of the ACM SIGMOD, Int. Conf. on Management of Data, 259 – 270. 1990.

25 Stonebracker, M. et al. "On Rules, Procedures, Caching and Views in Database Systems", Proceedings of the ACM SIGMOD, pp. 281 – 290, Atlantic City, 1990.

26 DeRose Steven J. "Xquery: a unified syntax for linking and querying XML." In proceedings Query Languages workshop (QL98), Boston Massachusen. Dec 1998.

27 Robie J. Lapp, "XML Query Language (XQL)." In proceedings Query Languages workshop (QL98), Boston Massachusen. Dec 1998.

28 Stavros Papastavrou, Panos K. Chrysanthis, George Samaras, Evaggelia Pitoura. "An evaluation of the Java-Based Approaches to Web Database Access". International Journal of Cooperative Information Systems, Vol. 10, Number 4, Dec 2001.