A Prototype Web-At-a-Glance System for Intelligent Information Retrieval

T.Catarci*, S.K.Chang#, L. B. Dong#, G.Santucci*
*Dipartimento di Informatica e Sistemistica Universit# di Roma "La Sapienza" Via Salaria, 113 - 00198 Roma - Italy, e-mail: [catarci/santucci]@infokit.dis.uniroma1.it

#Department of Computer Science University of Pittsburgh Pittsburgh - PA 15260, e-mail: [chang/dong]@cs.pitt.edu

Abstract: Web-at-a-Glance is a system to assist the user in creating a personalized database by gleaning the most relevant information from a web site or several web sites. This paper presents this new approach for intelligent information retrieval from web sites, and describes the prototyping of the WAG system as an active index system.

1. Introduction

With the rapid expansion of the wired and wireless networks, a large number of soft real-time, hard real-time and non-real-time sources of information need to be quickly processed, checked for consistency, structured and distributed to the various agencies and people involved in information handling. In addition to databases, it is also anticipated that numerous web sites on the World Wide Web will become rich sources of information [4, 12].

However, since too much information is available, information related to an important event could be missed because people are unable to track the manifestations of an unfolding event across multimedia sources over time. What is needed, from the information technology viewpoint, is an *Active Multimedia Information System (AMIS)* capable of processing and filtering multimedia information, checking for semantic consistency, discovering important events, and structuring the relevant information for distribution.

In [10] an approach for the human- and system-directed discovery, fusion and retrieval of multimedia information is described. This approach is based upon the observation that a significant event often manifests itself in different media over time. Therefore, if we can index such manifestations and dynamically link them to one another, then we can check for consistency and discover relevant information. This dynamic indexing technique is based upon the theory of active index [7]. Horizontal reasoning and vertical reasoning techniques can then be applied to consistency checking.

Information may come from both real-time and non-real-time sources. Real-time sources include sensors, cameras, etc., and non-real-time sources include both databases and web sites. Real-time sources usually generate continuous streams of data and the problem is to discover relevant events and to abstract the data into useful information. The approach described in [10] is a step towards that direction. To retrieve information from a well-structured database does not pose any major problem. However, to retrieve information from a web site with no predefined structure poses a problem. It is to this problem that we now turn our attention.

For intelligent information retrieval from the World Wide Web, in [3] we proposed the *WAG* (*W*eb-*A*t-a-*G*lance) system which can assist the user in creating a customized database by gleaning the relevant information from web sites containing multimedia data. WAG first interacts with the user to construct a customized conceptual view of the web pages pertinent to the user's interests, and then populates the database with information extracted from the web pages. In this paper we will describe the most important components of the WAG system - the Page Classifier and the Conceptualizer - and illustrate how the WAG system can be prototyped as an active index system. Since WAG can be realized as an active index system, it can be seamlessly integrated with the active multimedia information system AMIS, either by *incorporating* it as part of the active index system or by *substituting* it for the active index system. This would lead to a very powerful system for information discovery, fusion and retrieval.

This paper is organized as follows: The approach for intelligent information retrieval is presented in Section 2. WAG makes use of the intelligent Page Classifier and the Conceptualizer, which are described in Section 3 and Section 4, respectively. Section 5 presents the concept of active index and tele-action objects. Section 6 explains how the prototype WAG system can be implemented using the TAO_HTML interpretor and the IC_Manager, and Section 7 discusses further research.

2. The WAG Approach for Intelligent Information Retrieval

First we describe an idealized scenario for intelligent information retrieval:

> *A user poses a few queries to locate information of interest. The user may also navigate to the web site to select some objects of interest. After this preliminary interaction, when the user is inactive or away, an intelligent agent will attempt to build conceptual views containing the information relevant to the specific domains. The information, even if originally expressed in different formats, will be presented in a unified way. The views will be conceptualized and populated in two different phases: a) an* on-line phase *during browsing sessions of the user; b) an* off-line phase *by the agent, which will visit related sites applying the existing search engines to populate the database.*

In the above scenario, the intelligent agent will structure the information in the web sites and present the relevant information to the user or store the relevant information in a database whose conceptual view is constructed by the agent.

To accomplish the objective suggested by this scenario, we propose the *WAG* (*W*eb-*A*t-a-*G*lance) system which can assist the user in creating a customized database by gleaning the relevant information from a web site containing multimedia data. WAG performs this audacious task by first interacting with the user to construct a customized conceptual view of the web pages pertinent to the user's interests, and then populating the database with information extracted from the web pages.

The WAG system can be realized as an active index (see Section 5), which contains multiple index cells (ic's) to perform various tasks. The most important components (realized as index cells) of WAG are the *View Constructor ic*, the *Page Classifier ic*, and the *Conceptualizer ic*. Each ic will in turn activate other ic's to cooperatively accomplish the objective.

The View Constructor ic performs the following steps:

- o Page categorization
- o Class detection
- o Role detection (in the word "role" we include both the notion of attribute and relationship as intended in many semantic models)
- o Class/role population

WAG could be used by different users, for each of them the active index builds a "personal environment" containing the user's profile, the conceptual views of the domains and the corresponding knowledge bases (see Section 4). However, a user who just starts interacting with WAG is allowed to import and possibly merge the personal environments of previous users (unless marked as reserved) so as to take advantage of the information they already discovered. The new personal environment resulting from the importing operations can be modified, extended, or even rejected by its owner. A further possibility could be to ask the web sites the permission to be marked as visited by a WAG user, and add to them link(s) to the conceptual base(s) of the corresponding WAG user(s) (also in this case the users' authorization is mandatory).

In the following two sections we give a more detailed description of the two main ic's of WAG.

3. Web Page Classifier

The Page Classifier ic analyzes the structure of the pages and classifies them as belonging to certain predefined categories. The categories are differentiated based on the various contribution they can give to the subsequent conceptualization phase. For example, in an organization the home page of an individual will become an instance of some class, while the index page of the organization will be transformed into a conceptual subschema. A page containing a form will provide a sketch of the underlying relational database, and a page containing a map will become an instance of a map class, etc. Once the page taxonomy is defined, the Page Classifier must figure out specific subclasses of SGML DTD generating the various page categories, and for each new page to be included check which category it belongs to.

The starting point for designing the Page Classifier is to carry out a preliminary analysis of the kinds of owners of HTML pages on the web:

- a) Individuals (it is worth distinguishing between individuals affiliated to some parent organization, and unaffiliated individuals);
- b) Organizations (they can be further classified into: scientific, commercial, service-provider, information-provider, etc.);
- c) Search engines (e.g., Lycos and Harvest) and directory browsers (e.g., Yahoo and Internet Yellow Pages);
- d) Others, who have more "funny" pages like chat sites, etc. (these sites are out of the scope of the present work).

Pirolli et al. [13] presented a categorization technique of web pages, which is used to identify and rank particular kinds of web pages such as index pages and organization home pages. We have extended their work in order to come up with a particular page categorization capable of passing useful information to the Conceptualizer ic.

We define five page categories:

- o Organizational home page: these pages represent the entry point for different kinds of organizations and institutions;
- o Index: these pages contain a large number of links to navigate towards other (usually related) pages;
- o Personal home page: these pages belong to individuals, who may or may not be affiliated with some organization;
- o Document: these pages have the purpose of delivering specific information and, consequently, the percentage of outgoing links vs. the total page size is very low.
- o Map: these pages are documents whose information content is primarily a map or a collection of maps. Map is actually a sub-category of document.

The Classifier analyzes a page in order to categorize it and to figure out some suitable characteristics. There are two different kinds of analysis: the first one checks the syntactical structure of the page in order to verify the presence of HTML keywords which signal specific objects, i.e., lists, nested lists, forms, maps, tables, applets; the second one calculates the probability of the page to belong to each of the above five categories, exploiting some relevant properties of the page. The properties we take into account are: page size; number of local (i.e., coming from the same site) incoming links; number of outgoing links; frequency of access, which indicates how often the page has been visited; and depth of the children nodes reachable by that page.

In order to determine the probability for a page to belong to a certain category, depending on its properties we can follow two different approaches.

The first one relies on statistical techniques, under the assumption that a representative sample of web sites is available and that we know for each page the category it belongs to. It is possible, therefore, to compute for each property the distribution of its values for both the overall set of pages and for each page category. In doing that, the domains of the properties involved in the computation have been suitably partitioned in sub-intervals. Then, for each interval f_i of a property f, knowing the overall number of pages belonging to such an interval, say $N(f_i)$ and the number of pages of a certain category c belonging to the same interval, say $c(f_i)$, we can express the probability

for a page to belong to the category c if its property f shows a value belonging to f_i as $c(f_i)/N(f_i)$.

Moreover, under the quite reasonable assumption that the probabilities introduced above refer to statistically independent events, we can compute the overall probability for a page to belong to a certain category combining the contributions coming from the pertinent properties. More precisely, if there are k properties involved in determining the probability for a page h to belong to a category c, and $p(h,c,f_j)$, j=1..k, represents the contribution of the property f_j, the overall probability has the following expression:

$$P(h,c)=1-[(1-p(h,c,f_1)) \ldots (1-p(h,c,f_k))]$$

Note that if the global distribution of a site strongly differs from the distribution we got from the above sample, we have to introduce suitable adjustments in computing the generic p(h,c,f).

The second approach, suitable when a representative sample of web sites is not available, is based on the idea of expressing the overall probability for a page to belong to a specific category c, through a weighted formula of the form:

$$P(h,c) = (w_1 V_1 + \ldots + w_k V_k)/(w_1 + \ldots + w_k))$$

where $V_1,\ldots, V_k$ are the values of the relevant properties for the category c and $w_1,\ldots, w_k$, are weights calculated by a simple neural net, trained on a significant number of real examples.

The result of the categorization phase is a feature vector associated with the page. The vector contains the following fields:

- o Personal home page: probability for the page to be a personal home page (percentage);
- o Organizational home page: probability for the page to be an organizational home page (percentage);
- o Index; probability for the page to be an index (percentage);
- o Document: probability for the page to be a document (percentage);
- o Map: probability for the page to be a map (percentage);
- o Forms: number of forms contained in the page (numerical value);
- o Lists: number of simple lists contained in the page (numerical value);
- o Nested lists: number of nested lists contained in the page (numerical value);
- o Tables: number of tables contained in the page (numerical value);
- o Applets: number of applets contained in the page (numerical value).

The feature vector will be used by the Conceptualizer ic in constructing the classes and relations of the conceptual schema.

4. Knowledge-Based Conceptualizer

The knowledge-based Conceptualizer ic is the heart of the active index for WAG. It receives input messages from the ic's described above (which also include the user's suggestions) and try to build a (partial) conceptual schema from the HTML pages of a certain site (it could be also a domain or an IP network) and populate the schema with different kinds of instances (e.g., URL, tuples, objects, etc.) extracted from the site. It is worth noting that reduced amounts of data are replicated in some sort of materialized views, while large data sets, such as relational databases, map and image data, are referred by external pointers. Moreover, the Conceptualizer has the duty of maintaining the graph of the web pages related with the conceptualization process. The Conceptualizer relies on an object-based data model, equipped with powerful typing and classification services.

The Conceptualizer may work in two distinct modalities: on-line (during the phase of knowledge acquisition guided by the user) and off-line (when searching for other web sites which are relevant for a specific domain).

A. On-Line Interaction

This phase is started by the user whenever the user, during a browsing section, finds a site containing information of interest for a specific domain. Note that WAG records the pattern of the user selections, in order to replicate them during the off-line interaction.

The Conceptualizer receives from the Page Classifier the graph representing the link structure of the site HTML pages, plus, for each page, its feature vector. It also receives, from the user, the specification of the domain of interest.

The Conceptualizer searches for the domain of interest among the ones it already knows. Note that the Conceptualizer incrementally builds and maintains a hierarchy of knowledge bases (KBs). In particular, the top KB contains the specification (in terms of properties and admissible values) of the concepts which have universal validity, i.e., they do not mean different things depending on the domain, plus pointers to the local KBs for the concepts which could have an ambiguous meaning, i.e. whose meaning changes once changing the domain. The lower level KBs, associated with domains and sub-domains, contain the complete specifications of the concepts which are relevant for each domain, plus the specification of interschema *isa* links towards concepts belonging to KBs placed at higher level in the hierarchy.

If WAG knows already the domain, it shows the user the corresponding conceptual schema (for validation purpose) and, if the user's

validation gives a positive result, it starts the conceptualization process of the new site from the concepts contained in the conceptual schema it already has, taking advantage of the knowledge contained in the corresponding KB.

In this second case, WAG asks the user to provide a list of concepts (keywords) which the user expects to find in that site. Then, it starts the conceptualization phase by trying to figure out the schema classes from the HTML pages.

B. Off-Line Interaction

During this phase no user involvement is needed. Off-line interaction can be executed (on a specific domain or sub-domain) only after the domain KB has been built.

The basic idea is that WAG navigates through the web (possibly taking advantage of existing search engines) in order to locate sites containing concepts similar to those belonging to the domain KB. Once on a site, WAG first activates the Page Classifier ic to obtain the feature vectors associated with the pages, and then starts the page analysis, first trying to replicate the access pattern followed by the user (if available). The way in which the analysis is carried out is very similar to the one described above for the on-line interaction. The main difference is that user's inputs are substituted by accesses to WAG's KBs. Such KBs are updated during the off-line interaction only if the information to be added does not conflict with the one already stored. Also, the conceptual schema of the domain is updated and populated under the same condition. However, conflicting information and/or unsolved problems met by WAG during the off-line interaction could be brought to the user's attention and reconsidered with the user's help.

C. Search Strategy

WAG typically starts from the HTML page which is rated as the top candidate for being the organizational home page (if any). Indeed, the organizational home page (if present) contains special information and has to be treated differently from the others. Typically, it is the first one to be visited by the user, and gives a sort of high-level overview of the site content. WAG first analyzes the set of user's keyword to match them against the organizational home page content and try to individualize sub-portions of the site page tree that derive from single items belonging to the organizational home page (e.g., in an organizational home page for a university, there probably are keywords such as: faculty, course, student activities, etc. Each of these words is followed by a link to a more specific page, which is the root of a subtree). During this phase the user may concentrate the conceptualization process on a specific subset of the items listed on the organizational home page, thus restricting the domain of interest.

Then for each subtree, WAG adopts a simple analysis strategy, based on the idea of iterating the phases of class discovery and role discovery. The class discovery can be carried out by using two different methods: *sequence analysis* and *page analysis*. On the other hand, role discovery is based on *link analysis*.

The purpose of sequence analysis is to find regularities in the items composing a page. Sequence analysis can be applied only when special pages, corresponding to "candidate-classes", are available. Such pages are characterized by two features: a) they include sequences of items and b) (optional) they have an introductory part in which one or more of the user's keywords (including synonyms) appear. Their presence can be discovered by looking at the results of the classification process. Indeed, they should satisfy at least one of the following: 1) be classified as "index"; 2) contain simple or nested lists; or 3) contain tables.

Page analysis is used to identify sets of HTML pages which can be grouped together to form classes. This is done either 1) from scratch because the previous phases fail, or 2) starting from the clusters already discovered by the link analysis. In the first case, it is worth running a preliminary phase of purely topological graph analysis, suitable to identify articulation points in the graph of the pages to create subgraphs (see, e.g., [2]), and then searching for page clusters first inside the subgraphs. Actually, the topological analysis phase could be performed anyway by the Conceptualizer, to get further hints for the conceptualization process. Pages are analyzed searching for: 1) text similarities; and 2) presence of common keywords (especially user-specified keywords) or interesting subparts, where "interesting" means any paragraph or set of paragraphs which could be highlighted if they are strongly related with a specific user's keyword.

The purpose of link analysis is to identify the sets of individual links which are good candidates to become roles between classes. In the following we use the term *fiber* to indicate each individual link, while we use the word *link* to indicate the overall set of component fibers. Firstly for each page, the outgoing fibers are grouped in order to identify clusters that will eventually form strong links (see below). There are several criteria for forming such clusters: 1) the presence of common keywords used as anchor points (these are also matched against the user's keyword list); 2) the presence of special keywords and/or symbols used as anchor points (e.g., the word "back" or the backarrow icon); and 3) the similarity of the url path (e.g., all pointed objects are in the same directory). A strong link (between two classes) is a link having at least two of the following features: 1) it is composed by many fibers (wrt the number of instances of each participating class); 2) it has a dual reverse link which is strong; 3) it corresponds to a relationship between keywords explicitly indicated by the user; and 4) its component fibers relate instances belonging to either class through a single step path. Secondly, if two or more classes have already been discovered from the previous analysis phases, all possible pairs made out of them are analyzed first . More specifically, for each pair, all fibers starting from the instances of each participating class (which could be either HTML pages or items of a sequence) are clustered based on the above criteria, trying to identify strong links (which are candidate relationships) between the two classes. Then, the links starting from each class and pointing to something not yet classified as either class or instance of a class are in turn analyzed, trying to identify fibers composing strong links according to the definition above, and verify the corresponding clusters of pages. If no class has been identified yet, the search is carried on by analyzing all pages, trying to identify fibers that are components of strong links.

WAG always proceeds by iterating a two-step sequence , the output of one phase being the input for the other. It starts from sequence analysis whenever possible, i.e. when candidate-classes are available, and then apply link analysis, to be followed by sequence analysis or page analysis, and so forth until no new classes or relationships are added to the schema. Note that a class individualized through either sequence or page analysis, can be furtherly decomposed if link analysis verifies the presence of links involving not the whole class, but a specific subset (see [1] for a similar approach). When no candidate-class is singled out, WAG tries link analysis first, seeking for clusters of classes, and then applies page analysis.

5. Active Index and Tele-Action Objects

In our approach of human- and system-directed information discovery and fusion, the human can define index cells for the discovery of significant events. The system can generate additional index cells (using for instance neural networks) to monitor significant events. We now describe in detail the index cell, which is the fundamental building block of an active index. In Section 6, we will discuss how to prototype the WAG system as an active index system.

An *index cell* (ic) accepts input messages and performs some actions. It then posts an output message to a group of output index cells. Depending upon the internal state of the index cell and the input messages, the index cell can post different messages to different groups of output index cells. Therefore the connection between an index cell and its output cells is not static, but dynamic. This is the first characteristic of the index cell: *the interconnection among cells is dynamically changing.*

An index cell can be either *live* or *dead*. If the cell is in a special internal state called the *dead state*, it is considered dead. If the cell is in any other state, it is considered live. The entire collection of index cells, either live or dead, forms the *index cell base* (ICB). This index cell base ICB may consist of infinitely many cells, but the set of live cells is finite and forms the *active index* (IX). This is the second characteristic of the index cell: *only a finite number of cells are live at any time.*

When an index cell posts an output message to a group of output index cells, these output index cells are activated. If an output index cell is in a dead state, it will transit to the initial state and become a live cell, and its timer will be initialized (see below). On the other hand, if the output index cell is already a live cell, its current state will not be affected, but its timer will be re-initialized. This is the third characteristic of the index cell: *posting an output message to the output index cells will activate these cells.*

The output index cells, once activated, may or may not accept the posted output message. The first output index cell that accepts the output message will remove this message from the output list of the current cell. (In case of a race, the outcome is nondeterministic.) If no output index cell accepts the posted output message, this message will stay indefinitely in the output list of the current cell. This is the fourth characteristic of the index cell: *an index cell does not always accept the input messages.*

After its computation, the index cell may remain active (live) or de-activate itself (dead). An index cell may also become dead, if no other index cells (including itself) post messages to it. There is a built-in timer, and the cell will de-activate itself if the remaining time is used up before any message is received. This parameter - the time for the cell to remain live- is re-initialized each time it receives a new message and thus is once more activated. (Naturally, if this parameter is set to infinity, then the index cell becomes perennial and can remain live forever.) This is the fifth, and last, characteristic of the index cell: *a cell may become dead if it does not receive any message after a prespecified time.*

Although there can be many index cells, these cells may be all similar. For example, we may want to attach an index cell to an image, so that when a certain feature is detected, a message is sent to the index cell which will perform predetermined actions such as prefetching other images. If there are ten such images, then there can be ten such index cells, but they are all similar. These similar index cells can be specified by an *index cell type*, and the individual cells are the instances of the index cell type.

We developed a tool called the *IC_Builder*, which helps the designer construct index cell types using a graphical user interface [9, 11]. The activated index cells are managed by the *IC_Manager*, which can run on a Unix workstation or on any PC with Windows. To give index cells an external appearance, the cells can also be associated with multimedia objects, leading to Tele-Action Objects.

Tele-Action Objects [5] or TAOs are created by attaching knowledge structure (active index) to the multimedia object which is a complex object that comprises some combination of text, image, graphics, video, and audio objects. TAOs are valuable because they can improve the selective access and presentation of relevant multimedia information. In the Virtual Library BookMan [8], for example, each book or multimedia document is a TAO because the user can not only access the book, browse its table of contents, read its abstract, and decide whether to check it out, but also be informed about related books, or find out who has a similar interest in this subject. The user can indicate an intention by incrementally modifying the physical appearance of the TAO, usually with just a few clicks of the mouse.

TAO can be realized as TAO-enhanced html page (see Section 6). The physical appearance of a TAO is described by a multidimensional sentence [8]. The syntactic structure derived from this multidimensional sentence is a hypergraph, which also controls the TAO's dynamic multimedia presentation. The TAO also has a knowledge structure (the active index) that controls its event-driven or message-driven behavior. The multidimensional sentence may be location-sensitive, time-sensitive or content-sensitive. Thus, an incremental change in a TAO's physical appearance is an event that causes the active index to react. To summarize, the TAO's syntactic structure controls its presentation; and the knowledge structure its dynamic behavior.

A Tele-Action Object TAO has the following attributes: tao_name, tao_type, ic, p_part, links, where tao_name is the name of the TAO, tao_type is the media type of TAO such as image, text, audio, motion graphics, video or mixed, ic is the associated index cell, p_part is the physical part of TAO (the actual image, text, audio, motion graphics, video, or a multidimensional sentence for mixed media type), and link is the link to another TAO (there may be none or multiple links).

A TAO can have multiple links. A link has attributes link_type, link_rel and link_obj, where link_type is either relational (spatial or temporal) or structural (composed_of), link_rel is either the structural relation composed_of or a relational expression involving spatial operators [6] or temporal operators but not both, and link_obj is the linked TAO.

Whenever the physical part of TAO is changed, such as the modification of the image or the multidimensional sentence, message(s) are sent to the TAO(s) and associated index cell(s). Sometimes no specific index cell is associated with the TAO, i.e., the attribute ic is null, in which case message(s) are sent to all cells so that those who can respond to the message(s) may be activated. The change of physical appearance of a TAO may be due to (1) manual input, (2) external input, or (3) automatic input from the active index system. Thus a TAO can react to manual or external inputs, and perform actions and change its own appearance automatically. For example, the user clicks on a book TAO, and all related book TAOs change their color.

6. TAO_HTML Interpretor

To prototype the WAG system, each component of WAG can be realized as an ic associated with a TAO-enhanced html page. Given a TAO-enhanced html page, we can use an interpretor to read this page, abstract the necessary TAO data structure and generate the normal html page for the browser. Therefore no matter which browser is used, the application program can run if this TAO_HTML interpretor is installed in advance. This can give some security guarantee. The user can also choose a favorite browser. Furthermore if in the future HTML is out of fashion, the user just needs to update the interpretor and change it into another language. The other parts of application will not be affected.

In order to use TAO_HTML, or TAOML, to define a TAO, the data structure of a TAO is extended. A TAO has the following attributes: tao_name, tao_type, p_part, links, ics and sensitivity.

o 'tao_name' is the name of the TAO, which is a unique identifier of each TAO.

o 'tao_type' is the media type of TAO, such as image, text, audio, motion graphs, video or mixed. o 'p_part' is the physical part of TAO. To implement it in the context of TAO_HTML, 'p_part' here can be denoted by a template which indicates how a HTML page looks like. Templates are some independent HTML pages to define the fundamental display element and location arrangement. For example, if the TAO is of image type, the template will just contain a HTML statement to introduce an image. If the TAO is of mixed type, the template will define some common parts and leave some space to insert the elements that is specific to this TAO.

o 'links' is the link to another TAO. A link has attributes 'link_type', 'link_obj'. 'link_type' is either relational (spatial or temporal) or structural (COMPOSED OF). in the context of TAO_HTML, a spatial link describes visible relationship between sub_objects inside one mixed object. For example, a mixed tao1 contains an image TAO2 and a text TAO3, then TAO1 has spatial link with both TAO2 and TAO3. A temporal link usually refers to an invisible object which is not a display element, but its activation time is influenced by the other. A structural link relates one TAO with another dynamically via user input or external input. For example, the user clicks a button in TAO1 will invoke another page TAO2, then there's a structural link from TAO1 to TAO2.

o 'ic' is the associated index cell. The flag is "old" if the ic already exists, or "new" if the ic is to be created. The ic type, ic_id list, message type and message content can either be specified, or input by the user (indicated by a question mark in the input string). A corresponding HTML input form will be created so that the user can send the specified message to the ic's.

o 'sensitivity' indicates whether this object is location-sensitive, time- sensitive, content-sensitive or none-sensitive. Then the same object can have different appearance or different functionality according to the sensitivity. For example, if TAO1 is content-sensitive, it is red when being contained in TAO2 while it is green when being activated by TAO3 via a button. The detailed meaning of sensitivity should be defined by user according to the requirement of applications.

o 'database' specifies the database that this TAO can access and/or manipuate.

The formal definition of TAO_HTML language can be described in BNF form:

TAO_HTML ::= TAO_BODY

TAO_BODY ::= NAME_PART TYPE_PART P_PART LINK_PART IC_PART SENSI_PART DATA_PART

NAME_PART ::= "name"

TYPE_PART ::= TYPE_SET

TYPE_SET ::= [image, text, audio, motion_graph, video, mixed]

P_PART ::= "template_name"

LINK_PART ::= empty | LINK_BODY LINK_PART

LINK_BODY ::= name = "link_name", type = LINK_TYPE, obj = "link_obj"

LINK_TYPE ::= [spatial, temporal, structural]

IC_PART ::= empty | flag = FLAG ic_type = "input_string" ic_id_list = "input_string" message_type = "input_string" content= "input_string"

FLAG ::= [old, new]

SENSI_PART ::= empty | SENSITIVITY

SENSITIVITY ::= [location, content, time]

DATA_PART ::= empty | "database_name"

In the template of a TAO, in addition to the normal HTML tags and definitions, there's a special TAO tag for link relation with other TAOs. It is defined as:

"link_name"

The TAO_HTML Interpretor can now be presented in pseudo-codes:

```
procedure interpretor(char *TAOname)
{
```

```
  open TAO definition file
  call TAO_parser() to construct the
     TAO data structure TAO_struct
  call template_parser(TAO_struct)
     to output HTML file
}


procedure TAO_parser(file_handle,link_type)
{
  while (not end of file)
  {
    read one line from the file
    distinguish tag and get information
       and store in data structure
  }
}


procedure template_parser(TAO_structure)
{
  if IC_PART is specified, output HTML statements
     to create a form to accept user's input and
     send message to the ic's through IC_Manager
  if template file exists
     open template file
  while (not end of file)
  {
    read one line from the file
    if (not  tag)
      output html text
    else
    {
      get link_name from the  tag
      search in the TAO_structure with link_name
      if (a link structure is found with
         the same link_name)
      {
        get link_type and link_TAO_name
        switch (link_type)
          case structural:
            insert  link in template
               to link with link_TAO_name
   case spatial:
            call procedure interpretor(link_TAO_name)
               to insert template of link_TAO_name
      }
    }
  }
}
```

As mentioned in Section 2, the most important components (index cells) of WAG are the *View Constructor ic*, the *Page Classifier ic*, and the *Conceptualizer ic*. Each ic is associated with a TAO, so that we can define three TAOML pages - the View Constructor page, the Page Classifier page and the Conceptualizer page - to quickly realize the prototype WAG system. The TAO_HTML Interpretor translates the TAOML pages into HTML pages, so that the user interface is easily done through a web browser.

The prototyping of WAG as TAOML pages with active index cells has the added advantage that the user can easily enter information to experiment with the prototype WAG system. Moreover, whenever the user accesses an html page, an associated ic can be instantiated to collect information to be forwarded to the WAG ic's, so that flexible on-line interaction can be supported for the Conceptualizer and the View Constructor.

7. Discussion

As we envision it, in the future WAG may come in several different flavors: a light-weight WAG as an end-user tool to glean information from a single web site on a daily basis; a professional WAG to perform the same task albeit for several sites; and a designer's WAG as a system for the web designer to design/restructure a web site or several web sites. A prototype of the light-weight WAG is being implemented to test the feasibility of this approach. As explained in Section 6, this prototype is implemented using the TAO_HTML Interpretor and the IC_Manager. This prototype enables us to experiment with the WAG system, to improve the important algorithms for page classification, sequence analysis, page analysis and link analysis for class discovery and role discovery.

Since WAG is an active index system, it can be seamlessly integrated with the active multimedia information system AMIS, either by *incorporating* WAG as part of the active index system (in which case we have a general purpose AMIS capable of dealing with real-time sources, regular databases and web sites), or by *replacing* the active index system by WAG (in which case we have a special purpose AMIS for structuring and retrieving information from web sites). Both the general purpose AMIS and the special purpose AMIS will have many applications. Therefore, in our future research we will build a prototype AMIS with WAG as the intelligent database agent, so that we can deal with both general applications and web-specific applications in information discovery, fusion and retrieval.

References:

[1] C. Batini, S. Ceri, S.B. Navathe, "Conceptual Database Design", Benjamin Cumming Pub., 1992.

[2] R.A. Botafogo, B. Shneiderman, "Identifying Aggregates in Hypertext Structures", Proc. 3rd ACM Conf. on Hypertext, ACM Press, 1991.

[3] T. Catarci, S. K. Chang, D. Nardi and G. Santucci, "WAG: Web-At-a-Glance", Technical Report, University of Rome, Rome, Italy, February 1997.

[4] T.Catarci, "Interacting with Databases in the Global Information Infrastructure", IEEE Communications Magazine, First Part of the Feature Topic Issue on the Global Internet, May 1997, to appear.

[5] H. Chang, T. Hou, A. Hsu and S. K. Chang, "The Management and Applications of Tele-Action Objects", ACM Journal of Multimedia Systems, Springer Verlag, Volume 3, Issue 5-6, 1995, 204-216.

[6] S. K. Chang, Q. Shi and C. Yan,, "Iconic Indexing by 2D Strings", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No. 3, pages 413-428, May 1987.

[7] S. K. Chang, "Towards a Theory of Active Index", Journal of Visual Languages and Computing, Vol. 6, No. 1, March 1995, 101-118.

[8] S. K. Chang, "Extending Visual Languages for Multimedia", IEEE Multimedia Magazine, Fall 1996, Vol. 3, No. 3, 18-26.

[9] S. K. Chang, "Active Index for Content-Based Medical Image Retrieval", *Journal of Computerized Medical Imaging and Graphics*, Special Issue on Medical Image Databases (S. Wong and H. K. Huang, eds.), Elsevier Science Ltd., 1996, 219-229.

[10] S. K. Chang, "Information Discovery, Fusion and Retrieval by Dynamic Indexing", Tech Report, Univ. of Pittsburgh, February 1997.

[11] P. W. Chen, G. Barry and S. K. Chang, "A Smart WWW Page Model and its Application to On-Line Information Retrieval in Hyperspace", Proc. of Pacific Workshop on Distributed Multimedia Systems, DMS'96, Hong Kong, June 27-28, 1996, 220-227.

[12] N. Gershon, J.R. Brown (Eds.), "Special Report on Computer Graphics and Visualization in the Global Information Infrastructure", IEEE Computer Graphics and Applications, Vol. 16, No. :2, pages 60- 75, 1996.

[13] P. Pirolli, J. Pitkow, R. Rao, "Silk from a Sow's Ear: Extracting Usable Structures from the Web", Proc. of CHI'96, ACM Press, 1996.