WELLS; Web Enhanced Language Learning in *Language Teaching and Language Technology*

A.Allodi, D.A.Dokter and H.W.A.E.Kuipers

April 28, 1997

1 Computer Assisted Language Learning and the World-Wide Web

Research on the use of computers for educational purposes received a short period of attention in the early days of Artificial Intelligence. The disappointing results due to the technical difficulties of creating *practical* CALL applications however, led to a silent period, until the level of technology had developed to a stage where linguistic tools started to get sufficiently fast and reliable. The last two decades or so, a large number of applications have been developed, following this technological evolution.

Technological progress simultaneously led to major improvement of communication between computer systems. One area in the field of computer-to-computer communication is of particular interest to CALL, namely the enormous growth of the *World-Wide Web* (WWW or just: Web), and the development of a number of Web protocols and standards that have introduced the possibility of distributing documents and applications via the Web. The major contribution of the Web however, is not the distributional power, but the way in which this information can be accessed and viewed. The use of a graphical, coherent interface, and the use of hyperlinks that enable one to associatively browse through series of documents provide a user-friendly front end to the Internet. The Web enables any user that is connected, to exploit the available resources on virtually any site within a friendly environment, within relatively short time. This development significantly increases the potential scope of CALL. Language courses can be found on the Web, one can communicate with native speakers around the world, etc. It is no longer necessary to find a tutor or a course locally, which introduces the notion of *Distance Learning*. (For an overview, see [2].)

CALL applications come in various forms, in support of several different pedagogical ideas. Most of the earlier applications provided repetitive drills and exercises, while in a later stage the focus shifted to improvement of communicative skills of students. For an overview of these different ideas and the development of CALL during the last decades see [10]. Often, applications that are based on drills and exercises require supervision, that is, a tutor. Notwithstanding the possibility of computerized tutors, this line of applications is less easy portable to the Web, also since a large amount of bookkeeping for individual students is necessary in general. In this sense then, the Web stays on the level of intermediary between students and teachers, that is, a communication channel. WELLS is mainly a *communicative* CALL application, where the emphasis is on improving the practical communicative skills of the learner. The computer is used as a tool more than as a tutor; the application provides no drills, but empowers the user to use or understand language (as in a specific text) [10]. Also, the fact that the main factor in language learning for WELLS is the text that is used, which is chosen by the student himself, implies that the system does not require any supervision. This line of applications therefore, can not only be distributed, but also implemented on the Web, that is, made available like any other Web resource without restrictions for the user. The Web then, can be regarded a global database, instead of simply a communication channel.

The world wide availability of enormous amounts of information (documents, programs, etc.) in combination with the current state of the art in linguistic technology introduces the question *what* on-line information is publicly available and how it can support students in language learning. Clearly, it would be quite hard to obtain a course in Frisian in a Japanese book shop, but it would be rather easy in Friesland. With the growing coverage of any topic on the Web, the probability increases that one can obtain this kind of information anywhere. Often the only problem is that information is distributed and inconsistent (which of course is one of the main features of the Web) therefore not available in an integrated, fit-for-your-purpose format. For language learning there might not be a course in Frisian for example, but there might be an on-line dictionary Frisian-Japanese, a number of on-line texts and probably some other sources, which, when integrated into a single source of information could constitute a means of facilitating language learning. Being able to integrate different sources in a clear way would therefore be a major improvement for a "distant" language student. WELLS provides such an integrated source of information on text comprehension, but the architecture can be regarded a touchstone for any application with a similar use of Web resources. An important feature of WELLS is that it not only extensively *uses* Web resources, but also *is* a Web resource in itself. The program is accessible with any JavaScript compliant browser (for example, Netscape version 3.0), which will be regarded a standard in this paper. This means that anyone with an Internet connection and a browser is able to use WELLS, without having to download it and configure it. A nice side effect is that the software resides on a single location, minimalizing the maintainance problem of updating links and formats.

1.1 Glosser-R*u*G

WELLS is loosely based on, and partly intended as a feasibility study for a Web version of an existing application: Glosser-R*u*G. This application facilitates reading of French texts for Dutch speaking students, by providing a number of on-line sources of information on individual words of that text. Glosser-R*u*G can also be regarded a tool for readers that know a bit of French, but can not read it quickly or reliably due to the presence of a number of unknown words in the text [5]. The sources that are provided are: a dictionary, morphological analysis and examples of word use in especially collected corpora. The value of this line of applications for language learning is taken to be undisputed. Several studies have shown that reading texts significantly improves the learners vocabulary; even without the ability of dictionary lookup contextual information often enables a student to understand words and word usage [6]. A study by Mondria [8] shows that context helps not only in comprehension, but also in remembering words. WELLS approaches the same idea in a broader sense: it is not restricted to a single language and provides a wider range of information, but is subsequently less structured in displaying the information found. The sources that are exploited, are already existing as independent Web sites, such that the environment is far less controlled than the Glosser-R*u*G environment. Moreover, applications on the Web are obviously slower than similar applications that are processed locally.

2 The Web as a Distributed Repository of Information

To fully appreciate the use of the Web as a global database, this section will shortly introduce the main principles constituting the Web, based on a more thorough discussion in [9], which will be regarded an authorative body of knowledge concerning the Web, for the remainder of this article. A number of terms, necessary for a thorough comprehension of WELLS, will be exemplified.

The last decade has shown an enormous explosion of information, not only because of the generation of new data, but mainly because of the increasing availability of existing data. The World-Wide Web (WWW), officially described as a "wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents" [1], has proven to be able to provide users on computer networks with a consistent means of accessing a variety of media in a simplified way. Using popular interfaces to the Web, for instance Mosaic and Netscape, the Web project has changed the way people view and create information - it has created the first true global hypermedia network. The Web is primarily defined by four standards: URLs (Uniform Resource Locators), HTTP (Hypertext Transfer Protocol), HTML (Hypertext Markup Language) and CGI (Common Gateway Interface) [9]. These standards ensure the preservation of at least a minimal consistency of the Web, which is apart from these standards, a highly instable and inconsistent collection of documents [3].

A very important effect of the development of the Web, is an enormous decrease in space and time necessary for data acquisition and storage. In mere seconds a large corpus can be obtained from a site on the other side of the world, which makes *permanent* local storage of large quantities of data unnecessary. (The Electronic Highway is of course restricted in its bandwidth, which implies that at least intensively used sites have to be multiply stored.) In this respect the Web can be regarded a global database. (An interesting side-effect of using the Web as a distributed database is that access to different sources can be parallel. Several servers can be addressed at the same time, and will process the requests independently.) Some factors however, severely constrain the data-freeway. These factors mainly involve safety and inconsistency. An important safety-issue is that non-local use of computers has to be restricted to *reading* data. Other manipulations, *writing* data and *executing* programs apart from the ones specifically meant for Web interaction, cannot be granted to non-local users, since this would imply that everyone would be able to change (or *destroy*) data on remote systems; these manipulations obviously have to be conducted locally for every system. Inconsistency of the Web implies that to be able to fully exploit the Web, methods have to be devised to deal with numerous different document formats and the instability of Web links and locations. Another major question of exploiting the Web, is the legal problem of ownership of the material made available on the Web. The mere fact that something *is* available does not automatically imply that it is public property and can be used for any purpose. Questions arise as to what extent one can exploit data from other sites, and *if* it is used, can the data legally be reformatted? This issue is beyond the scope of this paper, WELLS being a scientific enterprise at this stage. Another issue is the speed of data transfer. Massive use of Web resources at the moment vastly increases processing time. Internet transfer speed however is constantly improved by technological development, so this problem will not be considered an *essential* constraint in this paper. To be able to freely manipulate the Web as a global database in *practical* applications, solutions have to be found to deal with the aforementioned restrictions.

2.1 Local vs. Remote

A frequently-used notion for introducing Web locations is the difference between *local* and *remote*. Both *local* and *remote* are always used to specify locations in relation to the machine, or user of that machine, that is focused upon. In general focus is upon whoever requests a Web transfer. In this paper whenever the focus is on the user, any other machine is remote, including the site where WELLS is located. When the site of WELLS itself is discussed, the users' machine can be regarded remote, as well as the informational sources WELLS exploits.

2.2 Client-Server Relations

An important relationship in the use of remote sites is the client-server relationship. The same principle of focus for local vs. remote holds for this relationship. The *client* is the local user, or actually a device on the machine of the local user, that requests an action by a remote machine, the *server*. A server itself however, can also be a client when it makes a request to another server. This is the case when a request by a user involves an action where the server needs information from another server. A user then, is always a client, but a transaction on the Web for a single user might involve several subsequent clients. (See section 3.1 for an example.) *Client* and *server* are therefore also relative to the transaction.

Once a request is sent to a server, control is given to this specific server. This means that a client can not interfere with the reply of the server until it has been returned to this client. The server therefore determines the precise format of the answer. At this stage the answer has become locally available and can therefore be manipulated. In general, however, the client is a graphical point-and-click user-interface, a Web-browser. The local machine is configured so as to directly display any replies in the browser, which implies that the remote server determines the layout of the requested data. This poses a problem for directly manipulating resources on the Web.

2.3 URLs

The *Uniform Resource Locator* (URL) standard is in principle just a convenient standard for addressing locations on the Web. The URL however not only specifies the site where the document thus addressed resides, but also what type of protocol is used to transfer it (FTP, Gopher HTML, etc.). Also a URL might include some information for dynamic pages, that is, pages that can appear in different forms according to the specific form of the URL. Example 2.1 shows a common URL. The first part (*http*) specifies the protocol, the second part (*//http://www.altavista.telia.com/cgi-bin/telia*) gives the address and the last part (*?country=nl&lang=nl*) specifies two variables with which to address this document.

Example 2.1
http://www.altavista.telia.com/cgi-bin/telia?country=nl&lang=nl

2.4 The Common Gateway Interface

The Common Gateway Interface (CGI) is a standard for interfacing with information servers, such as Web servers. CGI is a programming language independent standard that enables one to address a script or a program, instead of a document. This script is executed real-time on the server, upon addressing it. The address consists of the name of the script, and, in general, a number of parameters. This allows the developer of a Web site to add complete algorithms to a document, accounting for highly interactive communication between client and server.

Although CGI code is usually referred to as CGI-scripting, the code can be written in any programming language that can be executed on the local system, such as C, C++, PERL, TCL etc. The choice of which language to use in theory only depends on what is available on the system. If a programming language like C or Fortran is used, the program must be compiled.

2.5 Integrating Web Features

The different protocols and standards discussed previously introduces the issue how to integrate these facilities in a user-friendly environment. Creating a highly interactive application on the Web is obviously something more complex than creating a well-structured HTML page. Three main issues are involved:

1. **Local interaction:**
   as much as possible computation should be performed on a single side of the connection, either server or client-side, in order to avoid Web overload and long processing times.

2. **Remote communication:**
   cooperation between client and server(s) to carry out information retrieval operations, dynamic creation of Web-documents, etc.

3. **Cooperative processing:**
   communication between local and remote application-sharing data structures.

In order to implement these relations properly, a means to implement client side interaction is needed. This is provided by *JavaScript* (JS) [7]. JS is a script language that enhances standard HTML with some features to implement dynamic pages client-side. For complex functions that have to be performed server-side a set of CGI scripts is needed. CGI scripting can easily perform server-to-server communication and it embeds a high degree of data safety (only a priori defined data can be accessed, retrieved and stored [7], due to the (developer) predefined arguments a CGI script can handle). Also, CGI can, in principle, perform any function that can be implemented in any language, offering a powerful means of computation. JS however, allows most of the interactivity to be performed where it belongs to, according to the original Web philosophy: the client machine. Furthermore JS can:

- automatically recognize and communicate with local browser extensions;
- dynamically change the internal structure of a Web document according to user generated events (clicks, focuses etc.);
- perform local search and information retrieval;
- interact with server-sided CGI scripts;

It is important to notice that, while processes generated from such a scripting cooperation lead to large advantages in building Web-based applications, skills needed for programming JS remain very basic, due to the nature of scripting languages in general. On the one hand, this enables rapid prototyping and development, on the other this implies that the functionality of JS is rather restricted. Java [4] offers a more powerful means of implementing on the Web.

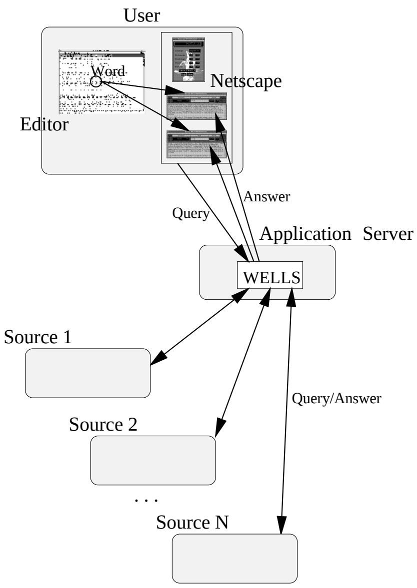

Figure 1: *The information flow of WELLS.*

3 WELLS on the Web

How can the enormous amount of information available be used in a way that fully exploits the advantages of the Web and avoids the dangers and restrictions? One of the most important means of organisation of the Web is clustering of related topics. Thematic homepages offer an overview of what is available on a specific topic and incorporate links to these sites. WELLS is a similar attempt of clustering thematical information. Based on a support program for language students, WELLS provides support for reading foreign texts by integrating different available sources on the Web into a single, unified source of information. This includes dictionaries, morphological analysis, thesauri, etc.

3.1 WELLS: Web Enhanced Language Learning System

To be able to fully appreciate the principle of WELLS, an overview of the information flow between the constituting sources has to be outlined. A graphical representation of this scheme is in figure 1. The first site in the scheme is introduced by the user, on a machine somewhere on the Web. This user controls a Web browser and a local text, constituting the words for which information is wanted. The second site is the site where WELLS is located. This component consists of all the code involved in acquiring information and formatting it into a specific form. This site therefore controls the front-end of the application. The last category in the informational flow is introduced by several sites supplying the different sources of information available within WELLS.

The location of the text has been chosen to be local to the user, to ensure that WELLS may be applied to any text the user needs to process, and to prevent any constraints on the use of WELLS concerning level of the user, etc. However, this also implies that WELLS cannot directly manipulate this text, since it resides on the client side. Words for which information is sought have to be explicitly sent to WELLS as a query. That is, either the client *types* a word in a query box, or the *cut-and-paste* option is used. The word is then sent to WELLS explicitly by the *submit* command, which is an obligatory step for this kind of requests on the Web. Also, the user is able to select the exact sources of information to be used, for example, a specific dictionary, if one is needed at all, etc. After the sources of WELLS have thus been specified, the application will send the query to the sites where this information is stored. In this transaction WELLS is a client itself, with respect to the sites where information is to be found, and a server with respect to the user. The addressed servers will return the requested data to WELLS, where the data have to be formatted before display, to ensure an optimal presentation towards the user. The complete unified information is now returned to the client, that is, displayed on the local screen.

The informational components of WELLS are:

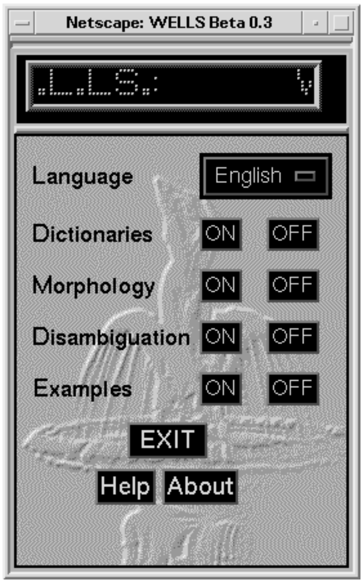

Figure 2: *The control panel of WELLS.*

- Dictionaries/Thesauri
- Morphological analysis
- Part-of-speech disambiguation
- Examples of word usage

All of these components potentially consist of a number of specific sources, enabling a user to choose between different instances of each informational category. The possibly large number of sources introduces the maintainance problem. The Web is highly instable concerning locations, layout etc. [3]. Therefore the maintainance of WELLS requires a continuous validation of the sources, with respect to format and location. However, one of the main advantages of shared applications shows here: since there is only one source of WELLS, new versions and modifications only have to be performed for this one source. Future research within the WELLS project will include automatic link updating and resource revalidation.

3.2 A Detailed Outline of WELLS

The front end of WELLS is constituted by the user-interface. The first part of the program that the user gets to see when WELLS is addressed[1] is a common Web page, containing four control buttons. (This page is not shown in this paper.) The button with the text "*ABOUT*" provides some general information about the project. "*HELP*" gives directions on how to use the interface, and "*CONTACT*" provides a link to the email address of the developers for suggestions, additional links etc.

START WELLS, starts the application itself. When the application is started, the Web browser disappears and the WELLS control window shown in figure 2 pops up. This window consists of a menu button for selecting a source language, that is, the language on which assistance is wanted, and four on/off switch pairs to specify the specific types of information to be displayed. Further, it contains another *ABOUT* and *HELP* button, and a button to exit the program. Upon exit, the browser reappears. When an informational source is switched on, a display window appears for the specified type. Every source window consists of a menu, where a user can choose a *specific* instance of a source type (for instance, a specific dictionary like Webster), a text display, and an entry window in which a word (or sentence, in the case of morphological analysis) can be entered. The *SEARCH* button submits this word to the server where the source is located and the *CLEAR* button clears the

[1] WELLS: *http://www.let.rug.nl/˜glosser/Wells/*

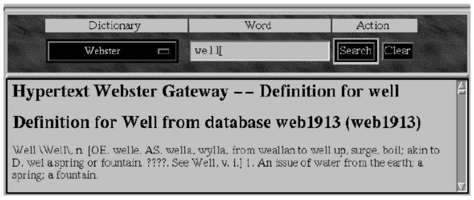

Figure 3: *The dictionary window where Websters' dictionary is selected and information from Webster is displayed about the word "well".*

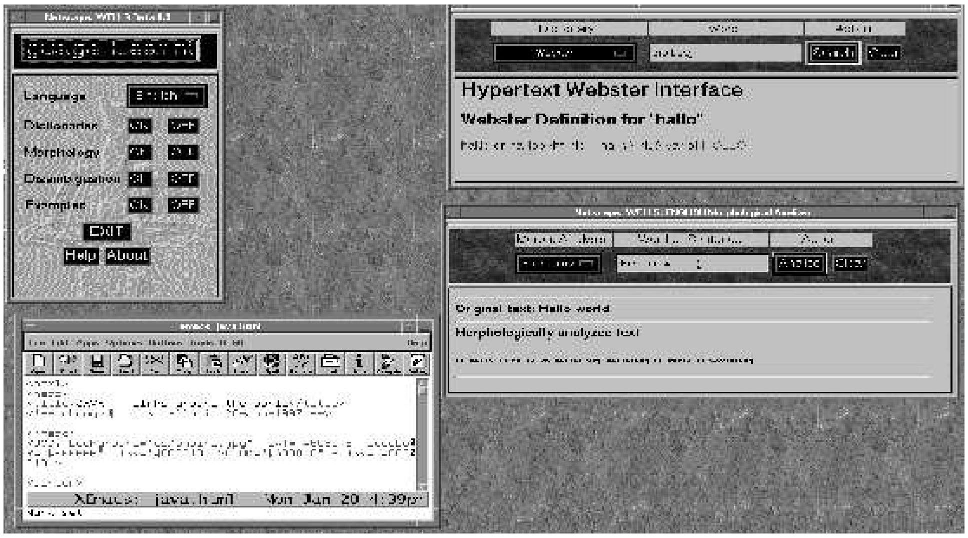

Figure 4: *The screen as the user sees it.*

entry window. After a query is submitted, the returned information is displayed in the text window. The dictionary window where the word "well" was submitted to Websters dictionary is displayed as an example in figure 3.

The user screen might look like figure 4 during the use of WELLS for support on text reading. (The precise appearance of course depends on the number of sources selected and the local configuration for the user.)

This constitutes the visible level of WELLS. The rest of this section is dedicated to a thorough discussion of the underlying level of informational flow.

The user-WELLS interaction starts upon the request for the welcoming page of the application. This is a common request by the HTTP protocol. Upon starting WELLS from this page, the dynamic interaction begins. The JavaScript code incorporated in the document is executed local to the client and accounts for all visible dynamics. When a user submits a request in one or more of the informational source windows, this request has to be sent to the URLs addressing the specific sources. However, the problem of reformatting the replies comes in here. In general it is impossible to intervene directly in the interaction between a client and a remote server, since the remote machine determines what the output of a request is going to look like. This poses a problem for WELLS, since some control over the output of remote servers is necessary to ensure optimal display of the retrieved information: in general, the HTTP protocol will return a full HTML-document, for example, a complete page on the Web browser with graphics, frames etc. An optimal integration for WELLS should at least account for the following issues:

- Only relevant information should be displayed
- The display space is limited; the original HTML documents retrieved are mostly quite large with respect to the displayed information.
- Some retrieved documents have interactivity features which interfere with the idea of WELLS: a unified source without the possibility of further browsing within the application. WELLS

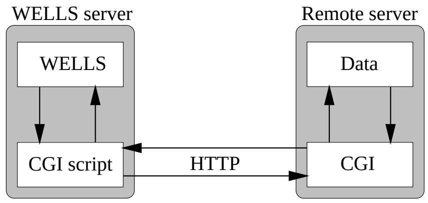

Figure 5: *The information flow between WELLS and a remote server.*

can be regarded a terminal station on the Web, no hyperlinks are provided in this application.

- A unified, consistent appearance of the different Web sources is preferred.

Therefore the interaction between WELLS and the remote machine cannot be direct. Several possible solutions exist for this problem. The simplest solution is to implement a routine that reformats the data at the moment it is received by the client, that is, the Web browser displays the data first before it is reformatted. This solution is quite inelegant, although the speed of the reformatting and the possibility of hiding of the output screen during first display makes it worth considering. Another, quite elegant but impractical solution is to cooperate with the servers providing the source of information in the sense that the servers can provide a specific layout for queries from an application. This would distribute the maintainance problem of this application to the informational sources, since the responsibility for the layout would be with the remote servers, that is, the provider, not the client. Clearly however, it would take some serious diplomatical effort to ensure such a cooperation. The solution implemented in WELLS is the application of a virtual client. This solution offers a very elegant means of Web communication. A virtual client, local to the application, can be addressed from the system, and subsequently send the query to the remote servers. The fact that the virtual client is regarded the real client implies that it will also receive the response. When the virtual client is a script instead of a browser, it can reformat the data and return it to the application. CGI scripts are a common means of implementing such a virtual client. The informational flow when using a virtual client is displayed in figure 5.

Thus instead of a direct connection between the client and the servers providing the requested information, the clients query is sent to a CGI script. This script creates the necessary query string (which is actually an URL) for each server and uses the standard UNIX utility *lynx* (a textual Web browser) to retrieve the documents represented by these URLs in text-only format. Lynx is used in a command-line mode; it just writes the retrieved information to a temporary file, which can be easily rearranged into a specified format before the CGI script returns the result to the application, thus before the user sees it. For each specific source a CGI script is written, since every source has its own lay-out specifications. The reformatted text is now simply returned to the application and displayed in the window from which the request was made.

This simple re-routing of information enables one to reformat sources on the Web into any appearance, thus to combine a number of sources into a consistent, integrated form. This offers the user a view on a number of normally distributed and incoherent sources, as a single application: WELLS.

4 Conclusions

WELLS exploits the Web as a database in a way that is surprisingly little used: incorporating distributed data into a thematic source of information, in this case a CALL application for facilitating the reading of texts. This idea is completely consistent with the general Internet philosophy of world-wide access to data, but takes the idea one step further. Integration of distributed data significantly improves the usefulness of the Web by providing a structured presentation of information on a specific topic, where otherwise one would have to search the Web extensively in a linear fashion to get the combined information, or would have to run several browsers in parallel. To be able to fully exploit the linguistic data available on the Web for language education, integration of information into a unified and structured source is an important step. Apart from integration, WELLS introduces another important idea. The program being a Web application itself, enables not only use of distant sources relevant for language learning, but also use of distant CALL programs (or programs in general). WELLS is still in an experimental stage; a number of issues has to be considered before a practical application will be within reach. Future work includes increasing the number languages WELLS facilitates, and consequently the number of sources that is being used, extending the types of information that is currently available, e.g. with syntactic analysis, semantic disambiguation etc., research of the possibility to automatically update links and information *about* sites and the development of a downloadable browser that uses resources on the Web without the necessity of a Web browser.

Acknowledgements

Part of this work took place within the Glosser-R*u*G project, at the department of Alfa-Informatica (Humanities Computing), University of Groningen. We would especially like to express our gratitude to Prof.Dr.Ir. John Nerbonne for supporting the WELLS project.

References

[1] A.Allodi and H.W.A.E.Kuipers. A guide to cyberspace. Technical report, URL: http://www.let.rug.nl/~hkuipers/cyberguide/, 1996.

[2] A. Allodi. Distance learning: Information needs and technology options. forthcoming, Alfa-informatica, University of Groningen, Groningen, The Netherlands, 1996.

[3] F.J.Burkowski G.V.Cormack C.L.A.Clarke and R.C.Good. A Global Search Architecture. Technical report, Department of Computer Science, University of Waterloo, Waterloo, Canada, 1995.

[4] Joy B. Gosling, J. and G. Steele. *The Java Language Specification.* Pre-Release Version 1.0, Draft 5.2 - July 3, http://crab.ccl.itri.org.tw/JavaDocuments/lang/, 1996.

[5] J.Nerbonne and P.Smit. Glosser-R*u*G: in Support of Reading. In *COLING96*, Groningen, The Netherlands, 1996.

[6] G. Krantz. *Learning Vocabulary in a Foreign Language; A Study in Reading Strategies.* PhD thesis, University of Göteborg, Göteborg, Sweden, 1990.

[7] J.J. Manger. *JavaScript Essentials.* McGraw-Hill, New York, NY, USA, 1996.

[8] J.-A. Mondria. *Vocabulaireverwerving in het vreemde-talenonderwijs; De effecten van context en raden op de retentie.* PhD thesis, University of Groningen, Groningen, The Netherlands, 1996.

[9] net.Genesis and D. Hall. *Build a Web Site; The Programmers Guide to Creating, Building, and Maintaining a Web Presence.* Prima Publishing, Rocklin, CA, USA, 1995.

[10] M. Warschauer. Computer-assisted language learning: An introduction. In S. Fotos, editor, *Multimedia language teaching*, pages 3–20. Logos International, Tokyo, 1996.