A GUIDED INTERFACE FOR WEB INTERACTION

Juan Falgueras, Antonio Carrillo, Daniel Dianes, Antonio Guevara
Dpto. Lenguajes y Ciencias de la Computación, Málaga University, Campus de Teatinos 29071, Málaga, Spain
jfalgueras@uma.es, carrillo@lcc.uma.es, ddianes@uma.es, guevara@lcc.uma.es

Keywords: User interfaces, Interaction styles, Human-Computer Interaction, Web applications, Web usability

Abstract: Many Web sites are for one use only and need to guide users through paths of decisions and data entering. Users use to feel insecure entering data in not well-known sites and don't mind at all being guided in their first try. *Goals Driven User Interfaces* (*GDUI*) are based on an interaction style specially conceived for this type of users (occasional users). The traditional method that still prevails in the field of Web design makes it difficult to create neither usable user interfaces nor *GDUI*. Modern paradigm of Web use through Web 2.0 applications encourages the development of this interaction style as can be seen in this paper.

1 INTRODUCTION

The evolution of programming standards and behaviour as well as the increasing processing power and speed of the communications has brought about the birth of a new type of application, the Web application. Most users consulting a Web site do so in order to perform one, or more, well defined objectives and wish to do so quickly. They have no user manuals or any other type of documentation. The user profile is difficult to determine a priori. On the other hand, Web applications are not wrapped and sold, instead, the software provided by the service is in a machine and the interaction interface is loaded onto the user's computer in a remote way.

Web applications can be executed on any computer, without any special software installation requirements and using the same personal document base, independently of the machine being used.

The applications may be viewed as Web pages but they have a much richer functionality with respect to the response to the user, which has been named the *Rich Client Interface*.

In traditional applications written for a specific operating system (or for the desktop applications), both the client and the server are in the same machine. This is the fundamental point that the web applications are changing. In such applications, the user's browser is a kind of "terminal" very different from the command terminal and capable of a high level of interaction:

- The browser can initiate a complex transaction using arbitrary data, communicating with the Server using AJAX. The importance of this should be noted, as before the existence of AJAX, transactions initiated by the client were limited to the sending of a Web form, with the inherent limitations.
- All the elements of the interface may be modified without the necessity for communicating with the server, using the DOM and the JavaScript programming language.

The main advantage of these two technologies (DOM and AJAX) is that of a direct and asynchronous communication, via arbitrary data structures, both server-client and client server; this is real bidirectional communication.

Using these technologies in the web applications increases the sensation of working with a desktop application, as it is unnecessary to call the server in order to make changes to the interface. An additional advantage is that the use of these technologies decreases the server load and the broadband necessities in the communication channel.

The Web applications in operation before the appearance of AJAX and DOM constitute an "abuse" of the basis of the WWW, as invented by

Tim Berners Lee, which was intended as a means of sharing hypertext documents linked via a computer network. Since the advent of these new technologies, giant steps forward have been made in the usability of what are becoming the real desktop applications via the Web.

Apart from the advantages obtained in relation to the ease of use and in the server time, the main advantage, and of particular interest to us in this work, is that AJAX and DOM help us to apply the studies in the field of HCI or Human-Computer Interaction (Dix, Finlay, Abowd, and Beale, 2003) to web applications.

2 THE USERS OF WEB APPLICATIONS

The Object-Action Syntactic-Semantic model proposed by Shneiderman (Shneiderman, 1980, 2005) suggests that user behaviour is based on two types of knowledge, *syntactic* knowledge about device-dependent details (e.g. *which* action erases a character, *which* action inserts a new line, *which* icon scrolls text forward, *which* abbreviations are permissible…), and *semantic* knowledge about concepts. The semantic knowledge is separated into task concepts (objects and actions) and computer concepts (object and actions).

By computer semantics, we are referring to common actions and objects, such as the knowledge that computers can keep information (action) in files (object). For users, the semantic knowledge of a task domain (for example the accounts in an accountancy program) can normally be associated to familiar concepts and should be relatively stable in the user's memory. A person can be an expert in the computer concepts, but a novice in the task concepts, and vice versa. Semantic knowledge is structured, device independent, and acquired by learning.

Based on this model, Shneiderman (Shneiderman, 2005), proposed three categories or types of users: *experts*, *intermittent users* and *new users*.

The first group of users are very well acquainted with the syntactic and semantic aspects of the system, and their objective is to complete their work quickly. They demand fast response times and to be able to execute actions with only a few strokes of the keyboard.

Intermittent users can handle the semantic knowledge about the task to be performed, and remember the computer concepts, but they have more difficulty dealing with syntactic knowledge/aspects of the specific applications.

Finally, it is assumed that the *novice users* have no experience working with computers and no syntactic knowledge about the use of the system and probably little semantic knowledge of computers in general. Among these users, there may be some degree of knowledge about the task domain (for example, accountancy knowledge) but others may have only a superficial understanding of it. These users may feel somewhat uncomfortable when faced with the prospect of having to use computers and this will get in the way of their using the system.

However, we are going to consider a fourth category of user: *occasional users (OU)*. These users need to use an application or operating system only occasionally, and in most cases, only once.

Although we could almost consider the OU as a novice, there is an important difference between the two: the OU is not interested in extending his/her interactive experience any further than that which is necessary to achieve his/her objective, and is not interested in becoming a frequent user or an expert. There can be no learning curve, as the result must be an immediate response to the user interaction at that particular time.

This is one of the largest groups of Web users. In general, the OU deliberately chooses not to or else have any time to practise, either learn or study how to use a system. Their main requisites are: being able to use it and not to have to spend time on learning to do so. For this, they will need ongoing assistance, and to be guided step by step, both in the process of carrying out the tasks and in the actual interaction with the system (i.e., in both syntactic and semantic knowledge). Normally such users refuse to spend any time on reading user manuals or even using the interactive help systems. This further strengthens the case for considering the web application user as an occasional user, since these applications are distributed without user manuals.

3 INTERACTION STYLES

Human-Computer Interaction or HCI (Dix, Finlay, Abowd and Beale, 2003,) (Lores, 2001) is the study of the interactive dialogue between people and computers. It analyses which dialogues are the most efficient, in the effort to minimize errors, increase user satisfaction by reducing their frustration, and therefore making the tasks involving people with computers more productive. It is a scientific research area related to Ergonomics, Psychology, and Software engineering, and includes various tools and

techniques, which help in the development of usable and good quality user interfaces (UI).

The very diverse *interaction styles* are frequently the subject of discussion in the HCI literature. For example, in the works of (Shneiderman, 2005), (Hix and Hartson, 1993), (Nielsen, 1993), (Preece, Rogers, Sharp, Benyon, Holland and Carey, 1994), (Ziegler, 1996), (Dix, Finlay, Abowd and Beale, 2003), both the types of user interfaces and the most common interaction styles were presented and categorized. Among which are the *Command Languages, Forms fill-in, Natural Language, Point and Click, Direct Manipulation, Menu, and Question and Answer*.

Our immediate objective is to look for the most appropriate UI for the OU, and, as we will see later, to examine whether its use and application are possible in the Web applications.

Because the application users are mainly OUs, the use of interfaces based on *Command languages* can be ruled out. Functionalities such as *Form filling*, the *natural language* and *point and click*, have their limitations and can only be used for simple or restricted task domains, or to complement another basic interaction style.

Currently, *Direct Manipulation* is the most common type of interaction in desktop applications. However, in spite of all of its advantages and its considerable functionality and usefulness, it is not necessarily intuitive or obvious enough for an OU. This is because these users would need to know or to learn the meaning of the visual representations, and the actions they can perform. Furthermore, in general these applications incorporate a large number of commands, menu options, toolbars and other independent semantic components, and normally these do not follow a hierarchical time structure which is suitable for the tasks or the objectives which the user is trying to accomplish at the specific moment of the interaction.

Probably, in most cases, *Menu selection*, and *Questions and answers* are the most useful to the OU, because they require only minimal expertise and guide users better than the alternative interactive styles. Nevertheless, these too have limitations as regards their functionality and usefulness.

In conclusion then, none of the traditional interaction styles have been specifically conceived for, nor are explicitly oriented to, the OU and therefore for a large portion of Web application users. Thus, in this work we present an interaction style conceived especially for the OU, which we have called *Goal Driven Interaction* (GDI) and we will analyze whether its use is possible in the Web applications.

4 GOAL DRIVEN INTERACTION

Goal Driven Interaction or *GDI* (Carrillo, Guevara and Gálvez, 2002) is a human-computer interaction style that is especially suitable for the type of interactive applications to be used by the OU.

This style, which has a conversational and sequential nature, can be considered a kind of combination of *Menus*, Direct manipulation and Wizards interaction styles. The aim is to guide, help and lead the user, step by step in a hierarchical and progressive way through the process of interacting with the application, based on the objectives and sub-objectives that the user has at a particular point. In order to accomplish these goals satisfactorily, the actions and tasks that must be carried out are described. All this must be done via a mechanism that holds simple and coherent at all times, using a simple UI with a well defined and organized structure (this will be described in section 5).

The goal is to simplify as much as possible the syntactic knowledge necessary to use the system and to provide the user both with the semantic domain concepts both of the task and of the computer so as to be able to perform the tasks successfully.

As a consequence, all the tasks and actions that the user must carry out, both internal and external to the application, must be specified in a hierarchical way, and in enough detail so that any potential user will be able to understand them and carry out the tasks correctly. The aim is to eliminate or reduce the possibility of making mistakes as much as possible. In any case, it will be necessary to establish and to offer mechanisms for rectifying mistakes, undoing actions, and cancelling goals already initiated.

The fundamentals of the GDI originate in the works of Newell and Simon (Newell and Simon, 1972) on the human reasoning mechanism for resolving problems. Their vision of problem resolution (as in GDI) was based on the breaking down or analysis of the main or general objective, into a hierarchical tree of sub-objectives, whose branch lengths would depend on the degree of subdivision within the sub-objectives. In the leaves of the tree, we would find fundamental sub-objectives reachable via basic information processes.

Based on this work, Card, Moran and Newell (1980, 1983), developed the most important of the existing cognitive models, the *Human Processor Model*. That starting paradigm (as in GDI) considers

the interaction process as a task of resolving problems. On the other hand, a psychological model of the humans is defined as consisting of three interactive systems: the perceptive, the motor and the cognitive, each one having its own memory and its own processor. The perceptive system is that which manages the external sensory stimuli, the motor system controls the actions, and the cognitive system provides the knowledge to be able to connect the two.

This vision of the user as an information processing system allows the formalization of the activities (physical and mental) involved in the task, and gave rise to some of the methods for modelling, specification and evaluation of the user interface that are widespread today, the GOMS (Goals, Operators, Methods, and Selection rules) methods (John and Kieras, 1994, 1996, 1996b). These allow the description of the sequences of behaviour and knowledge that the user needs to have in order to correctly interact with the system.

Therefore as the GOMS model corresponding to an interface gathers the knowledge users must have and the procedures they must follow, the aim of GDI is that the user should not have to dedicate time to acquiring this type of knowledge, but instead, the interface should provide it, at the same time as the interaction process is taking place. All this will be in accordance with the specification or the (GOMS type) model of the interface that the analyst will have developed in the analysis and design phase.

In order to accomplish the information gathering and the modelling task for this type of UI, we have devised a methodology and a specific notation based on NGOMSL (Kieras, 1997), which can be consulted in (Carrillo, Falgueras and Guevara, 2005).

Another important aspect is the possibility of using an automatic tool which, from the model or specification previously developed, will allow a prototype to be obtained quickly (even though it may be very basic), which will reflect the result of the modelling process. This tool will even allow tests to be done with users to check the suitability of the mental model of the users to the conceptual model of the interface. For this purpose, we have developed the GDIT tool (Carrillo, Falgueras and Guevara, 2006).

5 GOAL DRIVEN INTERFACES

As we will see in the next section, unlike what normally happens in the normal *WIMP interfaces*, a *Goal Driven User Interface* (*GDUI*), in its strictest version, does not include the standard application menu bar, nor the toolbar, nor the icons nor the quick access buttons for the main functions or commands, nor contextual menus, etc. Instead, in a GDUI, to reduce the complexity, different system commands or functions will be presented to the user (gradually and in a veiled way) while he/she is "browsing" the hierarchy of goals and actions being offered. This will happen in a specific area of the interface (the GDW), as the user initiates new sub-*tasks* or performs new *actions*.

For this purpose, and so that users are presented with a simple user interface with a well defined and organized structure and an interaction mechanism which is simple and consistent throughout the whole application, the GDUI will be structured in the following three parts (Fig.1): the *Goals Driver Window* (GDW), the *Working Window* (WW), and the Active Goals Hierarchy Bar.

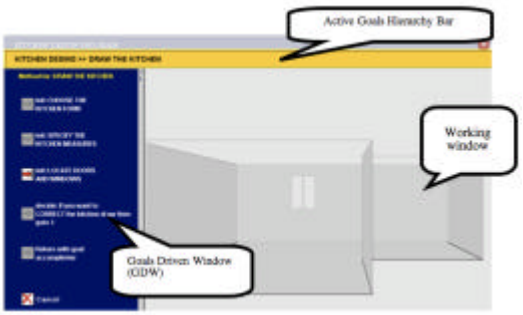

Figure 1: General structure of a GDUI

5.1 Goals Driver Window (GDW)

The GDW is the area of a GDUI (situated on the left hand side) where the main interaction takes place. This is the part of the interface where the user is guided and led by the hierarchy of objectives to be reached and the actions to be carried out, and where the different functions of the application can be accessed. It is a substitute (or an alternative) to the usual menus, access icons, toolbars or other typical elements of the direct manipulation WIMP type interfaces (as can be seen in Figure 1).

5.1.1 Methods and Selections

The GDW will begin with the *goal* of the highest level that can be reached with the application, the highest abstraction level and will always present and offer the user, either the *method* or the *selection*

which allows him/her to achieve the current *goal* at any given moment.

If the current *goal* can be reached by following a specific *method*, the GDW will show the set of *steps* that constitute the *method* (Figure 2, left). The GDW will always clearly indicate the next *step* to be taken, and the system will only allow *this step* to be performed.

The *steps* constituting a specific *method* may be one of the following:

a) The *step* may involve the initiation of a new *sub-goal* (which at the same time we have another *method* or another *selection* associated).
b) Or the *step* may require the user to perform a specific *action.*
c) A third type of *step* exists (which is mainly used for the repetition of sequences of steps, and for rectifying errors) which consists of the user *deciding* if he/she wishes to skip (within the *method*) to the specific step indicated, or to continue with the next one in the sequence.
d) Finally, all the methods will finish with an *step* which indicates that the *goal* has been accomplished and after which, the control flow will be returned to the next step after the previous method in the active hierarchy.

Figure 2: Example of GDW, of a GDUI, offering a *method* (left) and a *selection* (right)

Otherwise, if in order to reach a current objective the user must perform a *selection*, the GDW will show the different *alternatives* that the *selection* consists of, in such a way that the user can choose one (Figure 2, right). Although somewhat unusual here, the user may also be offered the usual choices existing in the direct manipulation WIMP type interfaces, although they are not actually in the GDW, but are in the WW. In this case, the GDW must provide the user with the *method* to be followed in order to perform the selection properly.

Regarding the basic user interaction mechanism in the GDW, if the current *step* of the *method* shown in the GDW (or the *alternative* to be chosen in a *selection*) involves initiating a new sub-*goal*, the user must click on the aforementioned *step* to initiate it. The GDW will then show the *method* or the *selection* that allows the said sub-*objective to* be accomplished. Otherwise, if the *step* requires an action or a task to be performed by the user, this must be carried out first. Once this has been done, the system must be informed (in the case where the system cannot detect it automatically) by clicking with the mouse on the corresponding *step* in the GDW.

5.1.2 Goal Cancellation

By default, all the *methods* and *selections* (or to be more precise, all the *goals* that can be accomplished following a *method* or performing a *selection*) can be cancelled by the user. As illustrated in Figure 1, and 2, this possibility can be seen in the interface by adding, after the last *step* of the *method* shown in the GDW (or after the last *alternative* if it is a *selection*) an additional and always active *step* that allows the user to request the cancellation of the aforementioned *method*.

Although other ways of cancellation may be considered, the one we propose by default is what we call *cancellation at method level (or high grade cancellation).* In this case, when a *method* (or a *selection*) is cancelled, independently of whichever *step* the user is at, the system will return to the *method* one level higher (i.e., the previous one) in the active hierarchy and the same state and *step* (within that previous *method*) which led him/her to the *method* (or *selection*) cancelled.

However, in some situations, the system will have to act on other external systems over which it does not have complete control. In these cases, for example when sending an email it would be impossible to "undo" an action or to cancel it (the email would already have been sent). In the hierarchy of objectives, a non-cancellable method represents a fixed node of no return for the rest of the interaction with the system. Obviously, if a not cancellable method is processed/ (OR a non cancellable method occurs), when the sub objective reaching the father method is returned, it cannot then be cancelled. These cases occur when it is impossible to go back to a previous state of the

system once this type of external action has taken place.

5.1.3 Forward and backward navigation

We have just presented the concept of *cancellable* and *not cancellable* goals. It is possible to go one step further and where appropriate, to "complicate" the interface and the interaction even more, in order to make it more flexible. This involves offering the user the possibility of "browsing" among the steps already performed (in the cases where these are reversible), i.e., going one step backwards or going forwards to the next step. This could be achieved in the interface by placing the corresponding buttons on the lower part of the GDW.

5.2 Work Window (WW)

The GDUI will assign most of the interface to what is called the work window (WW). In this part of the interface the visual representation of the work performed by the user and by the system will be presented, also the state of the task in progress, the state of the system, the representation of the objects of interest etc.

5.3 Active Goals Hierarchy Bar

This small area of the interface (normally situated in the upper region) is responsible for showing the hierarchy of goals at all times, i.e. the hierarchy of objectives that the user has followed from their initial goal to the present point in time.

6 APPLICATION OF THE GDI TO THE WEB

The traditional method that still prevails in the field of Web design make it difficult to create usable user interfaces for the applications. The fact that the logic of the application is separate from the interface and is connected by a network that does not guarantee a certain communication speed or stability, such as the Internet, means that interacting with a Web application is normally a continuous sequence of pages being updated.

Once the connection is established and depending on the connection speed, the server will eventually return the related content, which may have a lot, a little or nothing to do with the content actually on display. Here, to facilitate the process, a series of cache mechanisms come into play. These are established by the HTTP protocol and are specified using server-client communication headers. This method is an improvement on some situations, especially when the page that is the result of an update shares some elements with a previously loaded page.

Nevertheless, without the capacity to make changes in the application appearance as quickly as is possible in the desktop applications, a browser and the Internet cannot compete with these applications in spite of being accessible from any location.

There are two technologies that will solve this limitation, and which will allow the GDI to be used in Web applications. Their acronyms are DOM (*Document Object Model*) and AJAX (*Asynchronous JavaScript and XML*). Each of these has an important role to play.

6.1 DOM

The Document Objective Model (LePera, 2002) is a way of structuring documents downloaded onto a browser. An interesting feature is that this structure has properties that can be consulted and changed and methods to be invoked. In this way, everything that is uploaded at any given time in a browser may be changed. Elements may be displayed or hidden, the colour or type of letter may be changed, and even advanced animations may be performed without the need to refresh the whole page in the server or with a plug-in. Any value defined in a HTML document may be modified. This includes attribute values and label values. It is possible to go even one step further, changing the labels of a HTML document, reordering them, making them disappear, creating new ones, or even making a completely new document.

6.2 AJAX

Traditional communication, be it either by activating links or filling out and sending forms is synchronous (Fig.3). In other words, the browser blocks any attempt at communication or any update of the interface while it is awaiting the response to a request being processed. Therefore, there is a waiting period in which depending on the browser, the screen may or may not remain blank. In any case, once the server responds, the phenomena known as *updating or reloading the interface* occurs, including those elements already previously loaded and which have not changed from one page to the other.

AJAX (Garrett, 2005) breaks this barrier by introducing asynchronous communication (Fig.3).

Using this API, the programmer can perform asynchronous calls to the server without waiting for the response. While this is being done, the user can continue using the interface as normal (executing steps or carrying out selections in the GDW of a GDI interface, filling in data, unfolding lists, writing information, etc) and even making new asynchronous calls. When the response to the browser is available, a *callback* type function (similar to the Java *listener* style) will obtain the information.

The *callback* function is defined by the programmer who uses AJAX in the JavaScript language. With this function, the programmer has codes that are the result of a call to the server and these can be consulted in order to find out if the call has been successful, and also to obtain access to the results in either the text version or as a DOM fragment.

Here the two worlds of DOM and AJAX coincide. Where there is only a response from the server, nothing happens, the interface remains unchanged. *Callback* function completed their purpose and finishes when there is nothing to be done with the result.

However, it is possible to perform modifications on the DOM document based on what is returned by the server. If the format of the data transmitted is HTML, we may simply use the inner HTML property of some element of the document to change its content. The change will be effective immediately. Interestingly, all the other elements outside the element we change remain the same. We have managed to modify a specific aspect of the interface without needing to reload all the other elements that have nothing to do with the operation performed by the user.

These two technologies open the door to much localized changes in the interface which improve the user experience, and in particular, allow the GDI to be applied to Web applications in spite of the large number of modifications that this type of user interface requires.

In any case, evidently, Web applications will never be able to be as fast as those in the desktop application, because although with AJAX only essential data are transmitted (asynchronously), this transmission is done using the Internet and is therefore subject to variations in speed and even power failures in the system.

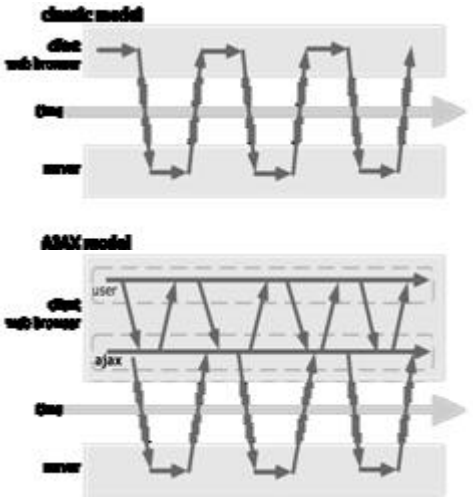

Figure 3: The synchronous interaction pattern of a traditional web application (top) compared with the asynchronous pattern of an Ajax application (bottom).

7 CONCLUSIONS

Although the initial objective of GDI was not to guide or lead the Web user (via objectives and tasks), its use and application to the pages and Web applications is as natural and effective that this might appear to have been the intention. In fact, the most frequent users of Web pages and many types of applications are occasional users. The most common type of Web interaction is precisely one without a learning curve and the frequency of use of many Web pages is relatively low. AJAX-DOM however covers perfectly the interactive necessities required to apply and use the GDI.

Furthermore, the maintenance of modified NGOMSL specifications required for the GDI specification (Carrillo, Falgueras, and Guevara, 2005), even allows dynamic building of interfaces from models on Apache servers for example, or of simple PHP interpreters of GDI language specification grammar.

GDI could be considered the ideal dialogue or interaction style to guide users through new and unknown tasks, especially for typical Web users who require quick results.

In relation to the representation in the *Work Window* of a GDUI, using AJAX allows us to directly update the content without having to act at the normal Web level.

Most of the bureaucratic procedures that have nowadays eliminated the use of paper, and can be

done via Web sites, can be easily performed via GDI type interfaces. The user of this type of bureaucratic and administrative task is, in most cases a perfect occasional user, without semantic knowledge of the steps to be followed in order to accomplish the necessary steps in this typical use of the Web.

REFERENCES

Card, S., Moran, T., Newell, A. 1980. *Computer text-editing: An information processing analysis for a routine cognitive skill.* Cognitive Psychology, 12, 32-74.

Card, S., Moran T., Newell A. 1983. *The Psychology of Human-Computer Interaction.* Hillsdale, New Jersey: Erlbaum.

Carrillo, A., Guevara, A., Gálvez, S. 2002. *Goal Driven Interaction.* III congreso Internacional Interacción Persona Ordenador, pp 68-75.

Carrillo, A., Falgueras, J., Guevara, A., 2005. *A notation for Goal Driven Interfaces specification.* Raquel Navarro-Prieto & Jesús Lores, Interacción 2004, Springer, Dordrecht, The Netherlands.

Carrillo, A., Falgueras, J., Guevara, A., 2006. *GDIT: Tool for the design, specification and generation of Goal Driven User Interfaces.* ICEIS'06.

Dix, A., Finlay, J., Abowd, G., Beale, R. 2003. *Human-Computer Interaction.* Second edition. Prentice hall.

Garrett, J.J. 2005. *Ajax: A New Approach to Web Applications.* http://www.adaptivepath.com/ideas/essays/archives/000385.php

Hix, D., Hartson, H 1993. *Developing user interfaces: ensuring usability through product & process.* New York, New York, United States of America. John Wiley & Sons.

John, B.E., Kieras, D.E. 1996. *Using GOMS for User Interface Design and Analysis: Wich Technique?* ACM Transactions on Computer-Human Interaction, Vol3, No.4, December 1996, pp 287-319.

John, BE. & Kieras, D.E. 1994. *The GOMS family of analysis techniques: Tools for design and evaluation.* Carnegie Mellon University School of Computer Science Technical Report No. CMU-CS-94-181. Also appears as the Human-Computer Interaction Institute Technical Report No. CMU-HCII-94-106.

John, B.E., Kieras, D.E. 1996. *The GOMS family of user interface analysis techniques: Comparison and contrast.* ACM Transactions on Computer-Human Interaction, Vol3, No.4, December 1996, pp 320-351.

Kieras, D. 1997. *A Guide to GOMS Model Usability. Evaluation using NGOMSL.* In M.Helander & T. Landauer (Eds.), The handbook of human-computer interaction. (Second Edition) Amsterdam: North-Holland, pp 733-766.

LePera, S.A. 2002. *Scripting For The 6.0 Browsers.* http://www.scottandrew.com/weblog/articles/dom_1

Lores, J. 2001. (Ed.): *Human Computer Interaction*. Libro electrónico editado por AIPO. Versiones disponibles en la red (http://griho.udl.es/ipo/) y en CD-ROM (ISBN Versión CD-ROM 84-607-2255-4).

Newell, A., Simon, H. 1972. *Human Problem Solving*. Prentice-Hall.

Nielsen, J. 1993. *Usability engineering.* San Diego, California, United States of America. Academic Press.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., Carey, T. 1994. *Human computer interaction*. Harlow, Essex, England. Addison Wesley Longman Limited.

Shneiderman, B. 1980. *Software Psychology: Human Factors in Computer and Information Systems.* Little, Brown, Boston, MA.

Shneiderman, B. 2005. *Designing the User Interface: Strategies for effective Human-Computer Interaction*. Four edition. Addison-Wesley Publishers.

Ziegler, J. 1996. *Interactive techniques.* ACM Computing Surveys, Vol. 28, No. 1, March.