Adaptive On-Line Information System

Serge Garlatti, Sébastien Iksal, P. Kervella

Adaptive On-Line Information System

S. Garlatti
IASC Laboratory, ENST Bretagne,
ZI de Kernevent, BP 832, 29285 BREST Cedex , France

And

S. Iksal
IASC Laboratory, ENST Bretagne,
ZI de Kernevent, BP 832, 29285 BREST Cedex , France

And

P. Kervella
Atlantide, Technopôle Brest Iroise,
Site du Vernis, CP n°2, 29608 Brest Cedex, France

ABSTRACT

SWAN project (Adaptive and Navigating Web Server) aims to design adaptive web servers for on-line multimedia information systems about nautical publications. It is a joined project between the IASC laboratory and Atlantide - a private company. The project is funded by the west region council and supported by the French naval hydrographic and oceanographic service. In on-line information systems, users used to only access a fragment of information space according to their current goal. User's goals can provide navigation support [1-8]. In our framework, user modelling is based on stereotypes and more precisely on prototypical user's tasks and on user's class. It also uses a domain model, an individual model and a maritime navigation context. The content and presentation adaptation is achieved by the sailor's class. The task model is used to design navigation processes and to define views of hyperspace. Some tasks determine the relevant domain concepts available in a particular geographical area, an adaptive method for them and a way to compute the hyperspace views. The system architecture is based on the one hand a WebObjects server [9] to manage user's recognition and web browser communication and on the other hand a knowledge based system managing the adaptation by means of a user's model [10].

Keywords : on-line information system, adaptive web server, user modelling, task model, domain model.

1. INTRODUCTION

Due to new technologies in telecommunication, most of information systems are available through Internet or Intranet. They are based on one or more web servers. They provide a new information retrieval mechanism based on browsing. Web servers supply with hypermedia tools for user-driven access to information. Nevertheless, hypermedia systems have some drawbacks : a user may become hopelessly lost in hyperspace when browsing in a large information space [11]. Then it is necessary to assist user's navigation for information retrieval. Reducing information space to access relevant information needed by users is a well known method to prevent from getting lost in hyperspace. In on-line information systems, users used to only access a fragment of information space according to their current goal. Goal analysis enables designers to specify relevant fragments and the navigation process. Then, user's goals are often used to design on-line information systems [1-8]. In general, goals provide navigation support.

SWAN project aims to design an adaptive web server for on-line information systems about nautical publications by means of user modelling. First of all, we present the SWAN project and its requirements. Secondly, the system's architecture is analysed and discussed. Thirdly, user modelling and task model is explained by means of a particular task example. Fourthly, the user's interface and the relationships between WebObjects server and the knowledge base is presented. In conclusion some perspectives are considered.

2. SWAN PROJECT

SWAN project (Adaptive and Navigating Web Server) is a joined project between the IASC laboratory and a private company called Atlantide. The project is funded

by the west region council and supported by the French naval hydrographic and oceanographic service. At present, sailors have to find out the relevant pieces of information in different categories of publications. All these publications are geographically organised around the notion of area which is recursively divided into sub-areas. These multimedia publications are composed of texts, photos, tables, drawings, charts and plans.

The on-line information system will provide together nautical information available in different types of publications - sailing directions, lists of lights and fog signals - for different categories of sailors and vessels to prepare a navigation or to navigate on oceans. In order to acquire users' goals and how they achieve them with paper publications, we made free and directed interviews of some different categories of sailors - military, shipping, commerce and yachtsman. This study showed that sailors have a common set of goals which are stables and are achieved in very similar way. Nautical publications for acquiring domain knowledge were investigated. In the first version of the web server, we decided to study these common goals, named services. Now, the architecture of the web server is presented.

3. ARCHITECTURE

Building an adaptive web server for on-line information system requires to design a domain model and a user model and an information base. The Web server has to recognise the user, the current stage according to the user's goal and maybe the corresponding application whether there are several applications running on the server. The HTTP protocol is a stateless protocol, that is to say, the server is unable to associate two queries from the same user and to link them. Consequently, a HTTP server is not sufficient to design a web server. HTML pages are generally static, then it is not possible to compute on the fly the content of a HTML page. Dynamic web servers require another tools.

Several tools are available to design dynamic Web servers, for example, web servers tools for databases. In general, the dynamic features of the web pages are limited to Information retrieval mechanisms of data bases: writing queries. WebObjects is a more opened tool for our purpose and provides an environment to produce Web applications [9].

WebObjects has three concepts : application, session and components. A web server may have several tools for different purposes. Each one is identified as an application in WebObjects which can store persistent states or manages persistent data. Sessions are periods during which one user is accessing an application. Different users may be accessing an application at the same time, a single application may have more than one session. There is one session per user on the Web server. An application is generally composed of a set of components. A component is a web page, or a portion of one, that has both content and behaviour. A component consists of three parts:

1. A template that specifies how the component looks. In other words, it is a generic HTML page. A template contains classic HTML tags and dynamic WebObjects elements.
2. Code that specifies how the component acts, that is to say how it has to respond to user queries
3. Binding that associate the component's template with its code.

Dynamic elements are the basic building blocks of a WebObjects application. They link an application's behaviour with the HTML page shown in the web browser and their contents are defined at runtime. Dynamic elements may be text, images, hyperlinks, conditions, ... By means of these dynamic element, it is possible to compute on the fly a Web page. The code part of a WebObjects application can be written in Webscript, objective-C, Java or mixed.

There are three possible strategies to state storage in a WebObjects application : in the server, in cookies or in custom state storage - external tools for instance. Then, the main problem is to choose a stage storage strategy for the user model, the domain model and the task model. We investigated two strategies: storage in the server by using Java to develop the different models and external storage in a knowledge based system. A good paradigm to develop the different models is objects. The object language has to provide some methods to access the models. In other words, it is necessary to browse the class hierarchy from top to bottom by means of the generalisation / specialisation relationship, but also over all relations. In Java, the « reflection » package enables us to browse each object or class and their relations. But, you need to know the class name and it is not possible to search the hierarchy from top to bottom.

In Artificial intelligence, the object-centered languages permit this procedures. Then, we chose the description logic Loom to manage the domain model, the user model and the task model. Loom is a description logic based on Common Lisp [10]. Loom has also a very convenient feature to develop a task model, a message sending protocol similar to those of object programming language. But it is more powerful and it is well integrated to the basic inference mechanism of description languages: the classifier. The data are stored in a data base. For the first version, we decided to put the data in Loom because it is easier, the amount of data is limited and Loom has a query language. For the reasons, the system's architecture is as follows (cf. fig 1):

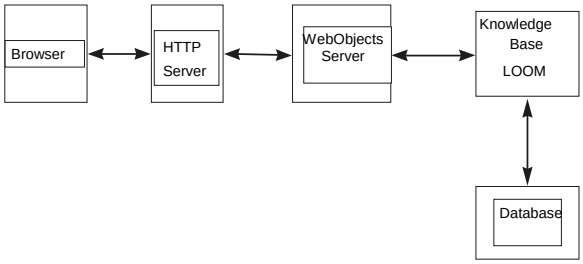

Fig. 1 System Architecture

WebObjects is connected with a HTTP server, and manages the following functions : computation of dynamic HTML pages, user identification, applications management, sessions management and events storage. WebObjects is the interface between the system and the user. Whereas Loom stores the different models. The domain model has two functions, firstly it is used as a reference for the adaptation and secondly it allows the data indexing. We use the query mechanism of Loom to search the useful data, because it filters the domain concepts. To manage the dialogue between Loom and WebObjects, we chose the socket protocol and a specific protocol language (lisp oriented).

As the user queries the system, the server gets the following information : the application, the session - i.e. the user -, the current Web page, the interaction type. It sends these data to the knowledge base server which retrieves the current task and by means of the different models computes the next step of the navigation process, that is to say the next component and its content. Loom gives these information to WebObjects which generates dynamically the next Web page for the right user. Now, we present the user model and the task model in details.

4. USER MODELLING

Stereotype, introduced by Rich [12], is an important element of user modelling and it has been extensively used because it gives a simple but powerful way for adaptation [13, 14]. In our framework, the user's model is based on stereotypes and more precisely on prototypical user's tasks and on a user's class.

The user's model is composed of a user's class, a task model and an individual model. Its structure is similar to the user's model of Hynecosum [2]. The user's class consists of a sailor's class and a vessel's class. The former has only one feature, the sailor category which can be professional or yachtsman. The vessel's class features are the following : length, breadth, height, tonnage, draught, navigation category which determines maximal distances from a shelter, vessel type (military, fishing, cargo, yacht, ..). The maritime navigation context consists of a set of navigation condition features : tide, time, weather forecast, general inference, GPS position (Lat/Long) or position chosen by the sailor. The user's individual model enables the sailor to choose an adaptation method for a particular task or to specify some parameters of an adaptation method and to choose the minimal depth of route.

According to Brusilosky, content-level and link-level, called respectively adaptive presentation and adaptive navigation support, are the two main classes of hypermedia adaptation [5]. The sailor's class is used for adaptive presentation and the task model for adaptive navigation. At present, the content and presentation adaptation is achieved in a simple way - for the first version : it depends on the sailor's category: professional or yachtsman ; sailing directions are different for these two user's classes. Adaptive presentation is processed in the same way whatever the task. Adaptive navigation support is achieved by means of a task model which uses the vessel's class, an individual model and the navigation context. Indeed, all tasks are available for each sailor's class. In a next version, we could design specific tasks for particular sailor's classes.

5. TASK MODEL

According to interviews, we find out four common goals for sailors - named Services - that are sufficiently general and high-level to be stable : route retrieval or creation, route information retrieval, port / anchorage, general information retrieval. Route retrieval or creation helps the sailor to find a route from a port/anchorage to another one. Route information retrieval provides navigation information, regulations, aids to navigation, lights, dangers to navigation, local conditions, currents, ... according to the route chosen. Port / anchorage gives to the sailor information about port entry, anchorage, marinas, facilities, services, etc., and a port retrieval based on the available services in the port and around. General information retrieval will provide history of weather, geography, oceanography, country and so on.

Task analysis of services showed that it is quite natural to represent them by a hierarchical task model [15]. The task model consists of tasks hierarchically organised by a composition relationships (cf. fig. 2). Tasks are divided into two classes abstract and atomic tasks. Abstract tasks are used to declare the navigation process. A control structure using standard operators - sequence (and) and selection (or) - achieves the sub-tasks ordering.

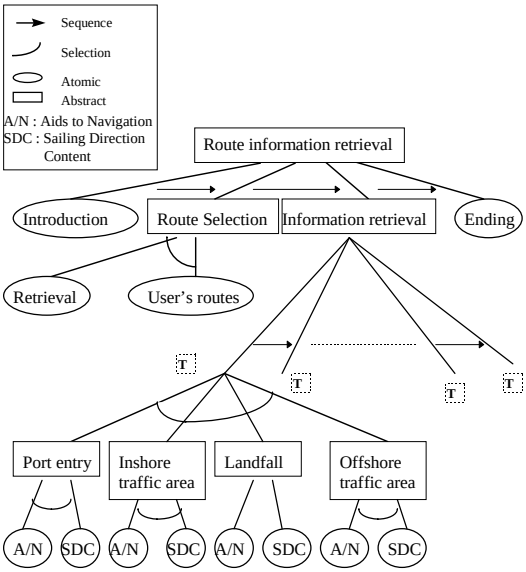

Fig. 2 Graph of the « Route Information Retrieval » service.

Each service begins with an « Introduction » task to explain the service's goals and a « Ending » task to close it, mainly for tutorial aspects of the first version. The « Route Selection » task is available to the sailor whether he has not previously chosen a route in the « Route Retrieval/Creation » service. Then, he can access the « Retrieval » sub-task to select a route. A route is composed of several route sections which are defined by two way-points, a compass course, a route section type (inshore traffic area, offshore traffic area, landfall, port entry), a minimal depth, a length, a sailing direction area, danger conditions. A route possesses some other attributes like: a departure and an arrival port/anchorage, a route category, a minimal depth, a length and advisable or not. After selecting a route, the sailor uses the « Information Retrieval » task to get relevant pieces of information corresponding to his route.

In our framework, two main information categories are provided to sailors: aids to navigation (buoys, lights, seamarks and alignments) and sailing direction content (texts, charts, images, drawings) which come from the sailing directions for professional or yachtsman and lists of lights and fog signals. This « Information Retrieval » task is composed of a sequence of sub-tasks, one task per route section type. A particular task class is associated with each route section type. Each class is composed of two sub-tasks, one per information category « aids to navigation » or « sailing direction content » to define the corresponding strategy for hyperspace views and adaptive method. Indeed, the two information categories are not processed in the same way because they are structured in a different way and are not accessed for the same reasons. Atomic tasks are used for information retrieval - all « aids to navigation » and « Sailing Direction content » tasks - and communication « Introduction » and « Ending » tasks. They are not composed of sub-tasks. A communication task gives some explanations to the user, specifies some user's needs and gathers data or information from users. An information retrieval task computes an hypermedia views allowing the user to browse in a small hyperspace. It determines the relevant sub-domain space, an adaptive navigation method and a way to compute hyperspace views according to particular sub-domain spaces.

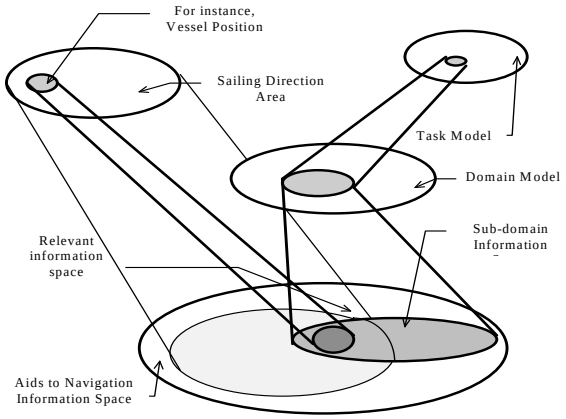

Fig 3. Hyperspace views from user's model.

In figure 3, the relationships between task model, domain model, sailing direction area are represented for an « information retrieval » task in the sub-domain space of aids to navigation. A sailing direction area can be represented by a polygon on a chart. For aids to navigation, a smaller polygon defined a priori or be the results of a computation is used to define the relevant information space. It is always included into a sailing direction area.

6. USER INTERFACE

Now, we go into details about the user interface and the relationships between WebObjects components and the knowledge based system. In a previous paragraph, a particular service, called Route Information Retrieval, was used to present the task model. We use it again to analyse the user interface and its principle. Indeed, adaptive systems are very useful to adapt the system to the user's needs - backgrounds, knowledge, goals, preferences, ... Adaptation leads to changes in the user's environment. Unless carefully designed, they may lead to an unpredictable and uncontrollable interface [3]. At present, we try to cope with these problems, we designed a user interface which have a static and stable part across all services and a dynamic part which gives transparency to the user. The user should see the system as a glass box within which the lower level components act as a black box [16]. As a first step to design such a glass box, we provide a stable environment to the user where the changes reflect some interior states of the system. In this paper, we only present the relationships between the user interface and the task and domain model. Indeed, they

are closely related to the user's goals and then we hope that they will be well understood by the sailors. Nevertheless, we need to check these points.

The user interface is presented in figure 4. For all services, the main window, having the entire web browser features, consists of four stable frames:

1. the top horizontal frame show the different available services,
2. the bottom horizontal frame gives the history of the current service,
3. the left vertical frame enables the user to select the pieces of information present in the chosen category - for instance, buoys, lights, seamarks, alignments, etc.,
4. the right vertical frame contains the selected piece of information.

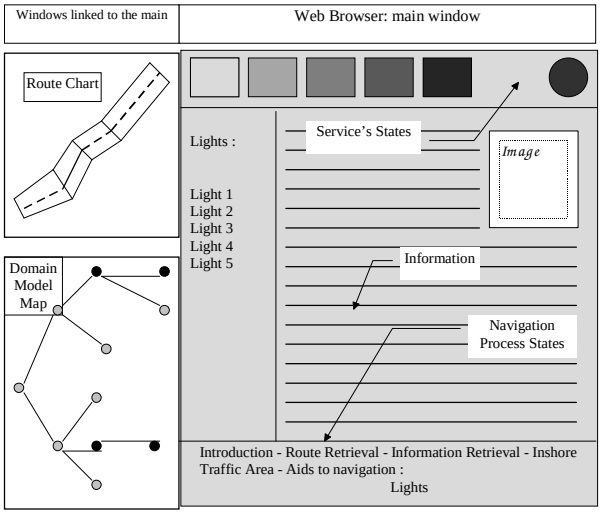

Fig. 4 User interface.

The left windows are linked to the main and are specific to the « route Information Retrieval » service. But, the main window is stable across the services. The top left window enables the sailor to choose a route section to get the corresponding pieces of information. A task determines the relevant sub-domain space, one or more sub-graphs of the domain model. The bottom left window offers a way to select a particular information category - for example, buoys, lights, seamarks or alignments in the category aids to navigation. The main changes that the sailor can perceive, are the following:

1. The service states, according to the corresponding task states, are displayed in the top horizontal frame,
2. The service history is display in the bottom horizontal frame and depends on the sub-task states of the current service and on the selected sub-tasks
3. The domain model map is an adaptive map in which the leaves are displayed in two colours - one for relevant categories and another for non-relevant categories.
4. The right vertical frame may display the next step of the current service

The main window behaviour is « static », that is to say stable over all services.

As shown, the task model is linked to the user interface and its adaptive features. But, it necessary to have another strong link between WebObjects and the task Model. Indeed, the knowledge based system establishes the next web page and its content. Then, it has also to determine the right WebObjects component for the current service step. In figure 4, the « Route Information retrieval » service is described. The main role of an atomic task is user communication and information retrieval and display. To cope with this goal, a description of one or more components and their features are associated to each atomic task. Only, atomic tasks have to communicate with the WebObjects server.

At present, the user and the domain's model are ready for the first prototype, the communication protocol between Loom and WebObjects is validated. Now, we are focusing our development on the WebObjects part - design of web pages - and the data recovery.

7. PERSPECTIVES

Some aspects of the current version of this on-line information system is intentionally simple. Indeed, all future users don't make a habit of utilising computer-based systems to achieve their daily tasks. Consequently, it is important to firstly design a software which is not too far from the current software in nautical domain. The purpose of the first version is also to show the benefits of adaptive on-line information systems to sailors, to acquire new goals for the systems and to suggest sailor propositions and comments before going further.

In the next stage, we plan to add some features in our adaptive web server : user's preferences for adapting the task model and the strategy to define hypermedia views, more sailor and vessels classes, new tasks, maybe specific to some user's classes. We thought about other features, but we need to evaluate the current version before modifying.

8. REFERENCES

[1] Waern, Y., *Cognitive Aspects of Computer Supported tasks*. 1986, New York: Wiley.

[2] Vassileva, J., *A Task-Centered Approach for user Modeling in a hypermedia Office Documentation system.* User Models and User Adapted Interaction, 1996(6): p. 185-223.

[3] Höök, K., *et al., A glass box approach to adaptive hypermedia.* User Models and User Adapted Interaction, 1996(6).

[4] Encarnação, L.M., *Adaptivity in graphical user interface: an experimental framework.* Computers & graphics, 1995. **19**(6): p. 873-884.

[5] Brusilovsky, P., *Methods and techniques of adaptive hypermedia.* User Modeling and User-Adapted Interaction, 1996. **6**(2-3): p. 87-129.

[6] Grunst, G., *Adaptive hypermedia for support systes*, in *Adaptive user interfaces: Principles and Practice*, M. Schneider-Hufschmidt, T. Küme, and U. Malinnowski, Editors. 1993, North-Holland: Amsterdam. p. 269-283.

[7] Tyler, S.W. and S. Treu, *An interface architecture to provide Adaptive Task-Specific Context for the user.* International Journal of Man, Machine and Studies, 1989(30): p. 303-327.

[8] Kaplan, C., J. Fenwick, and J. Chen, *Adaptive Hypertext Navigation Based on User Goals and Context.* User modelling and user adapted interaction, 1993. **3**(2): p. 193-220.

[9] Apple, *WebObjects*, . 1997, Apple Computer.

[10] MacGregor, R.M. *A Description Classifier for the Predicate Calculus*. in *Proceedings of the Twelfth National Conference on Artificial Intelligence*. 1994: p. 213-220,.

[11] Conklin, J., *Hypertext: An introduction and Survey*. IEEE Computer, 1987.

[12] Rich, E., *Stereotypes and user modeling*, in *user models in dialog systems*, A. Kobsa and W. Wahlster, Editors. 1989, Springer verlag: berlin. p. 35-51.

[13] Kobsa, A., *User modeling: Recent Work, Prospects and Hazards*, in *Adaptive User Interfaces: Principles and Practice*, M. Schneider-Hufschmidt, T. Kühme, and U. Malinowski, Editors. 1993, Notrh-Holland: Amsterdam.

[14] Kay, J. *Lies, Damned Lies and Stereotypes: Pragmatic approximations of users*. in *Conference on user modeling*. 1994: Hyannis, MA: p. 175-184.

[15] Rasmussen, J., A. Pejtersen, and L. Goodstein, *Cognitive system Engineering*. 1994, New York: John Wiley & Sons.

[16] Boulay, B.d., T. O'Shea, and J. Monk, *The Black Box inside the Glass Box: presenting Computing Concepts to Novices.* International Journal of Man-Machine Studies, 1981(14).