A Modular Software Platform for Programming Web Applications

Mauro Mosconi - Marco Porta

Dipartimento di Informatica e Sistemistica - Università di Pavia
Via Ferrata, 1 - 27100 - Pavia - Italy
mauro@vision.unipv.it - porta@vision.unipv.it

Abstract

Writing applications for digesting web information is a rather complex task. In order to let programmers easily build web-based applications we have developed a general purpose data-flow visual programming environment that can be effectively used to find, collect and manipulate information available on the web. The system was originally devised to help the user to develop applications in a totally controlled manner. However, the VIPERS environment, thanks to its modularity and its flexibility, allows the easy integration of software modules that can even be characterized by autonomous intelligent behavior, thus allowing simple implementation of agent-based web applications. We therefore propose our system as a useful platform for rapid prototyping and experimenting with software agents.

1. Gathering Really Useful Content from the Web

In the present network age, individuals have become able to freely access a huge amount of heterogeneous information available on the World Wide Web. Global information is continually growing, raising the problem of how to search efficiently for what one is most interested in.

Search engines represented the first solution to this problem. By accessing large databases created by special programs roaming the Internet, they answer the user's keyword-based queries by providing indexes (links) to the relevant sites found. A different approach is represented by *personalization services* (i.e. PointCast), where relevant information customized for each user is delivered constantly.

Both services are very useful and popular, but they can not completely satisfy the needs of the most demanding users. As far as the search services are concerned, in fact, users (with fairly clear objectives) may want to accomplish more complex tasks while being freed from repetitive browsing tasks. In contrast, personalization services may require too much effort from the users who must specify their preferences in details.

Actively gathering information and passively receiving information are complementary activities: since Internet users are not required to be skilled at using computers, researchers and software companies have invested great efforts especially in the latter direction, towards the development of the so-called software agents, which know users' interests and can act autonomously in their behalf, by digesting large amounts of information. This is a promising approach but can not be considered a panacea: it is evident that many web-related tasks raise a challenge for automation but do not require the complexity involved by an adaptive intelligent system, which must be carefully trained and refined.

At the basis of our research, lies the convinction that for a large class of applications, non-naive users could accomplish their goals better and more rapidly by directly composing simple programs themselves, while preserving a sense of control over the system. Moreover, our experience indicates that such programs can be easily and conveniently expressed within a data-flow visual programming environment, as we will show in this paper.

Since we are aware that for the majority of Internet users programming has to give way to some form of delegation, our research is now moving towards more complex and challenging goals, involving autonomous and possibly cooperating agents. Our paper discusses the possibility of using a visual programming environment as a modular platform for rapidly prototyping web applications and to experiment with agents.

2. A Data-Flow, Visual Approach for Web-based Applications

The system we use for building web-based applications is VIPERS (Bernini and Mosconi 1994), a general-purpose visual programming environment based on an augmented data-flow model and developed at the University of Pavia.

VIPERS uses a single interpretive language, Tcl (Ousterhout 1994), to define the elementary functional blocks (the nodes of the data-flow graph). Each block corresponds to a Tcl command (or procedure): such a command may itself invoke the execution of other UNIX programs as subprocesses.

This paper has been published in the Proceedings of the *First International Workshop on Interaction Agents*, May 24, 1998, L'Aquila, Italy, pp. 61-63.

According to the data-flow model, VIPERS programs are graphs in which data tokens travel along arcs between nodes (modules) that transform data tokens themselves. As shown in Figure 1, VIPERS elementary modules have a square shape and present connection points, or ports, on their lateral sides. Programs are assembled by direct manipulation, by positioning and properly connecting the needed modules. Entire programs may therefore be built without typing any line of code: after having picked from a certain window the needed functions (the nodes of the graphs) which are represented by special icons, the programmer simply drags and drops them into the main workspace. A couple of simple mouse-clicks are then sufficient to create an arc between two nodes.

During program editing, typing efforts and errors are minimized. Even syntactical errors are avoided, since the graphics editor is able to prevent the user from positioning or connecting the blocks in the wrong manner. The debugging activity consists of examining the program graph on the display. The particular box-line representation allows users to easily insert proper viewing monitors at various points of the program to show partial results. Documenting programs does not require great additional efforts, and even other non-programmers or occasional users are able to understand such a "visual" code.

2.1 A Personalized Newspaper on the WWW

The main idea behind this paper is that collecting and manipulating information available on the web is an application susceptible to a data-flow approach. Very concisely, what is generally needed is the capability of accessing web pages, looking for keywords, selecting and following interesting links, extracting useful information and possibly reformatting it. Some of these tasks may be very complex, especially when they are based on A.I. techniques, as for intelligent agents.

Nevertheless, once they have been made available as *black boxes*, data transformations and filtering can be conveniently expressed even by unskilled users by means of a data-flow language, provided by proper iteration constructs and procedural abstraction mechanisms (Hils 1992).

As a practical example, we present here a simple web application which accesses the main headlines page of an online newspaper and identifies titles containing at least one among some given keywords (see Figure 1). Each headline is arranged in a distinct row and has a link to the corresponding article. The purpose of the application is to download the pages relative to the selected titles, so that they can be quickly accessed later on. Moreover, an HTML page (index) reporting the list of all the picked headlines, each one having a link to the corresponding piece of news file, is then created and displayed in a web browser. The application in itself is simple, but allows some potentialities of web-based programs for visual composition to be highlighted. Once a library of elementary components has been created, in fact, it is very easy to achieve even conceptually complex functionalities.

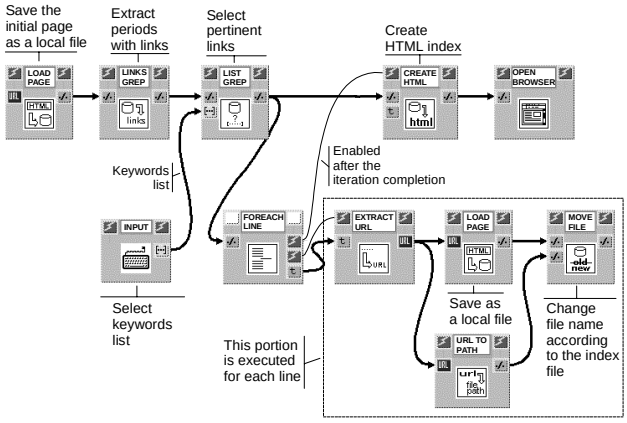

Figure 1: a simple web-based application within the VIPERS environment. In the program, modules characterized by an autonomous behavior may also be inserted.

Each input/output port in VIPERS is characterized by a special icon indicating the correspondent data type. Data types used in the example are: URL (Uniform Resource Locator), •/• (file path), [...] (list) and t (text).

Referring to Figure 1, block LOAD PAGE connects to an URL (received as an input or defined as a default) and downloads the corresponding HTML page on a temporary file, whose path name is made available as its output. Such a file is analyzed by block LINKS GREP, which extracts only periods containing links. The function of block LIST GREP is to examine these periods, in order to elicit those containing at least one of the keywords present in its input list. The keywords list may be requested at every execution of the program (in the example such task is accomplished by block INPUT) or it may be specified as a default for block LIST GREP. The path name of the file holding only the selected lines is then provided both to block FOREACH LINE and block CREATE HTML.

The *foreach* control construct, in VIPERS, allows elements of a sequential data structure (in this case, a file) to be analyzed and made available, one after the other. Such a compact block might be replaced with an explicit cyclic subgraph. Block FOREACH LINE emits the various lines of its input file, as they are taken out, but only at the end of this process does it activate the control signal enabling block CREATE HTML. In the meanwhile, block EXTRACT URL extracts the URL it incorporates from every input line and block LOAD PAGE downloads on a temporary file the page addressed by the just extracted URL, whose name is properly settled later on.

When all the lines have been scanned by block FOREACH LINE, block CREATE HTML is activated and generates an HTML file to be used as an index for the locally stored pages. Lastly, block OPEN BROWSER displays the created HTML file in a web browser, reporting the list of the news items which were searched for, accessible through links to local files.

This is just the kernel of the program we daily use to check for news about our favorite soccer team: other blocks (not shown here) are used to calculate the first URL

(depending on the date), to organize downloaded pages into a special archive and to automatically send some articles via e-mail after a further filtering.

2.2 Integrating Agents into a Modular Architecture

The VIPERS structure is completely modular: that is, every elementary block composing the application data-flow graph may be characterized either by an autonomous, intelligent and adaptive behavior or by a totally predetermined behavior. Whole applications, possibly independently developed (for instance, within Lisp environments), may easily be integrated with the system, provided that they can be invoked through a UNIX command line specifying the input parameters.

For example, a researcher may focus on the functioning of a single agent, by exploiting those languages and software systems which are the most suitable for the task being examined, and then, through the VIPERS support, he/she may easily test functioning within applications which require some knowledge of the Internet protocols, by using and combining predefined black-boxes.

However, once an agent has been made available in the form of a VIPERS block, a programmer user wishing to build a custom fully controlled application may decide to use autonomously behaving elements as well, perhaps to integrate the results obtained with those achieved in parallel through a more procedural approach.

In the abovementioned example, if the programmer knows exactly what the goal is (for example, to find articles about a specific soccer team), the system turns out to be useful and properly dimensioned. If, however, the study of user behavior is required, to understand his/her preferences, the system lends itself to the integration of new modules, which may be represented as blocks in the program graph. For instance, these modules may include a WWW browser in which there is an interaction agent that monitors user behavior, a learning engine that infers user preferences from the interaction agent, and a scoring engine for the selection of the proper articles for the personalized newspaper (Kamba, Sakagami and Koseki 1997).

3. Conclusion

So far, in our research activity we have been focusing on making the process of building web applications easier by means of direct manipulation techniques. Very often, in fact, even complex problems can be properly visually abstracted so that they can be easily and progressively tackled by letting users specify exactly what they want through a visual programming approach. This does not mean that the users have to take responsibility for every move the computer makes. The VIPERS system, thanks to its modularity and its flexibility, allows the easy integration of software modules characterized by autonomous intelligent behavior, allowing easy implementation of agent-based web applications. We believe that our system may also constitute a useful platform for rapid prototyping and experimenting with agents.

References

IEEE Internet Computing, vol. 1, n. 4, Jul-Aug 1997.

Bernini, M.; and Mosconi, M. 1994. VIPERS: a Data Flow Visual Programming Environment Based on the Tcl Language. In *Proceedings of the AVI'94 International Conference*: ACM Press.

Hils, D. D. 1992. Visual Languages and Computing Survey: Data Flow Visual Programming Languages. *Journal of Visual Languages and Computing*, vol. 3, 69-101.

Kamba, T.; Sakagami, H.; and Koseki, Y. 1997. ANATAGONOMY: A Personalized Newspaper on the WWW. *International Journal of Human-Computer Studies*, vol. 46, n. 6, 789-803.

Ousterhout, J. 1994. *Tcl and the Tk Toolkit*. Addison Wesley.