Programming Web-Based Applications within a Data-Flow VL

Roberto Idini - 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

This paper shows how a general purpose data flow visual programming environment can be effectively used to find, collect and manipulate information available on the web. The discussion highlights the elementary functions that are needed to implement such applications, as well as the language control structures that can make program development easier. Starting from the analysis of a simple application, some advice is given about the potential of this approach. We also explain how we intend to make a suitable portion of the programming environment available through a common web browser.

1. Introduction

Since the Web emerged as a global information network, people have tended to spend a growing proportion of their lives in front of computer screens, browsing, selecting, collecting and manipulating information. An important attempt at employing computers and networks effectively is represented by the so-called software *agents* [1], which know users' interests and can act autonomously in their behalf, digesting large amount of information while freeing users from repetitive browsing tasks.

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.

Our theory is 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.

2. The Programming Environment

The system we use for building web-based applications is VIPERS [2], a general-purpose visual programming environment based on an augmented data-flow model and developed at the University of Pavia.

VIPERS uses a simple interpretive language (Tcl) 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.

3. A Data-Flow Approach for Web-based Applications

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. Data transformations and filterings can be conveniently expressed even by unskilled users by means of a data-flow language, provided with proper iteration constructs and procedural abstraction mechanisms [3].

As a practical example, we present here a simple web application which accesses the main headlines page of an on-line 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, a 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.

Copyright 1998 IEEE. Published in the Proceedings of VL'98, 1-4 September 1998 at Nova Scotia, Canada. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works, must be obtained from the IEEE. Contact: Manager, Copyrights and Permissions / IEEE Service Center / 445 Hoes Lane / P.O. Box 1331 / Piscataway, NJ 08855-1331, USA. Telephone: +Intl. 732-562-3966.

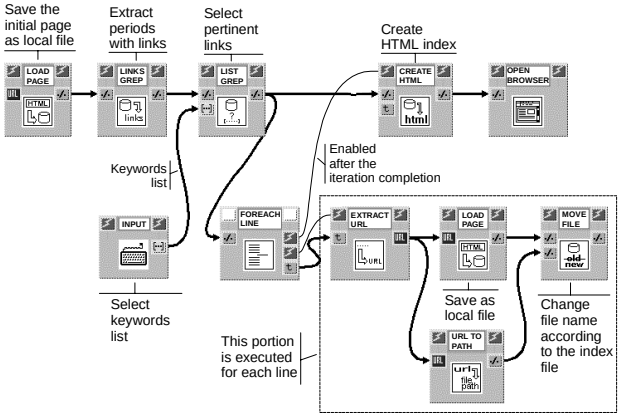

Figure 1: a web-based application example

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 favourite 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.

4. Conclusion

Data-flow visual programming lends itself to employment by people who are unskilled at using computers, such as generic users of internet services often are, and so it proves to be especially convenient even for complex behaviors. The purpose of the tool we are developing is simply to allow the common web user to build his/her custom agent-like applications in an easy manner.

It is important to stress the fact that block functionalities can be defined and extended as one likes, both textually (through the Tcl scripting language) and visually (by using ready-made blocks). Therefore, it is easy to create a large functions library supporting the various programming requirements in the web applications area.

Our future aim is to make an on-line portion of the described visual environment available, to function as a server. That is, any user, by just connecting to the proper site through an ordinary browser provided with the Tcl plugin (free and downloadable), will be able to exploit a vast ready-made blocks library and visually build his/her own application. Program execution will occur remotely, on the server. As the user prefers, results can be displayed in the client browser or forwarded via e-mail to a specified address.

References

[1] *IEEE Internet Computing*, Vol 1, Num 4, IEEE Computer Society, Jul-Aug 97.

[2] Bernini, M., Mosconi, M., "Vipers: a data Flow Visual Programming Environment Based on the Tcl Language", in *Proc.AVI'94*, ACM Press , 1994.

[3] Hils, D. D., "Visual Languages and Computing Survey: Data Flow Visual Programming Languages", *Journal of Visual Languages and Computing*, Vol. 3, 1992.