SOFTWARE DEVELOPMENT IN A WEB-BASED ENVIRONMENT

Lukito Edi Nugroho[1]

ABSTRACT

The advances of computer networks and data communications have led us to an era of network-based software. A very popular approach in developing such software is building the software on top of Web technologies. The capabilities of working under heterogenous platforms and seamless integration of static and dynamic information make Web application programs very different to those of ordinary, non-Web applications. Consequently, developing such a software needs a different approach. A research has been conducted to investigate the differences in their development issues. The research was performed by comparing the development methods of an application aimed to run in the two different environments. This paper reports the results and suggests some points on migrating an ordinary software to a Web-based environment. Understanding this is useful as the global computing tendency is shifting to distributed computing where differences and details in underlying hardware and operating systems are abstracted out of the users.

1. INTRODUCTION

The needs of acquiring more computation power, sharing resources, and ease of personal communications motivate the advent of networked computing. The Internet has provided the largest medium for this type of computing. Among Internet applications, World Wide Web (WWW or Web) is the most popular one. Invented by Tim Berners-Lee from the CERN Laboratory in Switzerland as an internal information exchange tool (Segal, 1995), Web has now been adopted as a main vehicle to run applications in a networked environment.

What makes Web gain its popularity is their three characteristics :

1. Enabling easy information access throughout a network, be it a LAN or the Internet. Information contained in files located in different geographical locations can be reached by following *links* or *references* that point to those files.
2. Uniform graphical user interface (GUI).
3. Providing mechanism to work under heterogenous hardware and operating systems through standardized protocols.

It has not been until recently when effort was made to extend Web capability to be able to deliver *computed*, instead of static, information. Based on certain user inputs, the information is computed by programs embedded into Web pages. The computation part is usually trivial and does not become the focus of the Web-based application. However, it is completely possible to promote the computation part to a more complex degree.

It has been understood that developing a non-trivial software needs rigorous software engineering methods that are applied throughout a series of steps (Pressman, 1992). These methods, however, are influenced by the maturity of software and hardware technology, which is constantly changing. It is also understandable that such methods are applied mostly to the development of applications that run on environments where heterogenity is not an issue (for example, stand-alone applications). A relevant question would then raise : is it appropriate to develop non-trivial Web-based application software using current software engineering methods ?

2. METHODOLOGY

The research was conducted by developing a non-trivial application using two different approaches. The application's goal is to control an office's equipment inventory and its dynamics (movements, repairs, throw-outs, usage authorization, etc.). A same set of software requirements was used in both approaches.

The first approach used visual design and programming under Windows 3.11 operating system with Borland Delphi 1.0 (Pacheco and Teixeira, 1995) as the development tool. Database management was performed by ODBC drivers provided by Delphi. This approach represents an 'ordinary' approach, for which a standard set of current software engineering methods are usually applied. The result is a standalone software that runs in a uniform environment.

In the second approach, the application was developed as a Web-based software using HyperText Markup Language (HTML) specification (McGuirk, 1996, Raggett et.al., 1997) to build the Web pages. PHP/FI (Lerdorf, 1997) was used as the Common Gateway Interface (CGI) programming tool (NCSA, 1996). Finally, Netscape Navigator was used as the browser. The development was performed under Linux operating system. Database management was performed by Mini SQL (Hughes, 1997). The Java language (Cornell and Horstmann, 1996) were once tried, although not used in the final implementation.

The experiences in using the two approaches are discussed below.

3. DISCUSSIONS

3.1. Basic paradigm

Visual programming is a type of programming where user interface design and implementation are put into focus. A wealth of dialog mechanisms are provided,

enabling programmers to build almost any form of user - software contacts. In Windows environment, visual programming is often related to *event-driven* programming. Event-driven is the underlying Windows programming paradigm. When a user action is originated, a message (event) is sent to objects that are responsible to react. This mechanism, however, is completely hidden from programmers in visual programming. Here, programmers start with user interface framework and moves into details by filling *procedural slots*, which usually implement activities that must be performed as reaction of occuring events. In other words, programmers define how an object should react for events that are sent to it.

CGI programming can be classified as ordinary procedural programming, where execution control mechanism such as conditional branching and loop control are heavily used to implement the application's algorithm. CGI programming is not an independent entity; a CGI program is strongly coupled to and cannot be separated from a Web page. When a Web page is visited, a CGI program linked to it is executed to produce certain information that will be output in the page.

It can be stated that both types of programming are not related and thus incomparable. However, as a user interface design mechanism, visual programming does have relationship with Web page design. It is possible to design (program) a Web page visually, as long as it produces code that complies to the HTML specification. This is what visual Web design tools have done.

3.2. Design issues

A Web page can accomodate information in a broader scope than ordinary applications through links and mechanisms that stand behind them (CGI programs, Java applets, and processing of various file formats). From the perspective of software design, this also broadens the scope of the design activities. Program (code) design, which forms the whole part in ordinary application design, is now only a part of a Web-based application design. The other part is the information that will be incorporated into the Web pages. The analysis and design of information content, its representation, and sequence presentation must be performed carefully. This often involves social and psychological aspects of human-computer interaction, whose importance is less in ordinary program design.

As an emerging technology, Web-based program design still lacks of well-defined methodologies. Like our experiences with software engineering in the past that the maturity of design methodologies was driven by programming methodologies, Web-based program design is heavily influenced by its programming languages, notably the C (Kernighan and Ritchie, 1988) and Perl (Wall and Schwartz, 1991) languages, which are procedural-oriented. Well-known architectural (general) software design methodologies that fall into this category, such as structured design (Pressman, 1992), can be applied.

Being confined to procedural-oriented methodologies makes Web not optimum for applications where procedural approach is not the best choice (e.q. simulations, expert

systems). Some effort has been done to enable other approaches. An example is the Internet Inter-ORB Protocol (IIOP) specification from then Object Management Group (OMG, 1996). The IIOP enables different Object Request Brokers (ORBs) to communicate. ORB is an object-oriented mechanism for software *objects* to communicate to perform client/server operations. For many cases, object-oriented is more powerful and flexible than procedural oriented (Booch, 1991). Another tool that provides object-oriented approach is the Java language (Cornell and Horstmann, 1996), which is designed specifically to build programs in heterogenous networked environment. However, both still have limitations. IIOP is not mature yet, while Java is not designed to replace C++, which is very popular in non-Web development environment.

In detailed design level, different interpretation on the 'module' concept causes another problem. In an ordinary application, a module is the unit of a software component, which is usually implemented in functions or procedures. One or more modules can reside in a single file. In a Web-based application, the modules are the Web pages, each resides in a single file. There is a fundamental difference in viewing what a module is. A function or procedure defines logical task that should be performed by the module, while a Web page is more oriented to user dialog sessions in using the application. Modularity measures such as cohesion and coupling cannot be applied easily to Web pages. For example, if a logical task comprises a series of user inputs, then it is not possible to modularize it into a single cohesive Web page.

Still related to modularity, the idea of page reusability (as opposed to function or procedure reusability) is also difficult to realize. There is no way to expose the public part of a Web page (for example, its parameters), so unless access to code in its parent page is granted, it is impossible to link the children page from another page. A consequential disadvantage is its inability to deal with similar tasks. A completely new page must be built for a similar task. A related problem is that Web application does not recognize 'global variables'. This makes very impractical to propagate a global value to all CGI programs in the application.

3.3. Implementation issues

What makes a Web-based application interesting is its natural ability to work in a heterogenous environment. This is very supportive if one has to build an application that must be able to run in different hardware and operating systems. In a Web application, there is no single code that must be inserted to deal with the heterogenous platforms. On the other hand, it is not possible to run a Windows program outside the Windows environment.

According to the experiment, there is no significant differences in effort in programming the user interface. However, in Windows environment, this is achieved by the use of a visual development tool (Delphi) that greatly reduce the complexities of user interface programming. Otherwise, programming user interface in Web will be much

easier. Regarding the result, the Windows-version application has more types of dialog forms, making the user interface more attractive.

Basically every programming languages that can output string to the standard output (stdout in C's term) can be used in CGI programming. Some languages are more difficult to learn that common general programming languages like C and Pascal. Perl language, for example, is notorious for its cryptic syntax, but is extremely powerful and compact to solve string manipulation problems, which almost always occurs in CGI programming. The others, such as PHP/FI, are easier and more practical than C or Pascal, but lacks of language structure to support complex computation. The choice depends on what aspect the developer put a stress on.

3.4. Migrating existing software to Web-based environment

Based on the experiment, the following points on migrating existing non-Web applications to Web-based environment are proposed :

1. Applications that are not based on procedural-oriented approach are advised not to be ported. It is possible to migrate object-oriented applications using Java language, but the full capability and flexibility of conventional object-oriented languages, such as C++, cannot be fully exploited. In non-Web applications, C++ has a full control of programming environment. For example, it can use its libraries, work with files, and run another programs. Java, on the other hand, cannot perform these for security reasons. When Java runs in Web environment as applets, these capabilities are disabled, because they can form holes for security breach and illegal accesses. For example, in Unix systems, system information of a remote host can be deduced from local files in the remote host.
2. Current Web protocol and programming tools are only suitable to be used to build Web-based applications that do relatively simple tasks. Applications that heavily use shared (common) modules cannot be ported without any sacrifice in its design. Related functions or procedures in an existing application, which are possible stored in different files, must be modified and rearranged so that they are grouped into the same Web page. This must be done in order to reduce unneccessary 'parameter passings' that travel accross several, possibly irrelevant, pages. This, off course, has the cost of duplicating functions or procedures in several Web pages.
3. Applications that requires advanced user dialogs that based on event-driven mechanism cannot be ported to Web environment. For example, it is common in Windows environment to have a drag-and-drop mechanism. This type of dialog is impossible to be implemented in Web environment, due to the inability of Web to work with event-driven mechanism applied to Web pages. However, simple event-drivent dialogs can be emulated by using frames (McGuirk, 1996). Frames can be used to divide a Web page into several independent subpages. These subpages can be used to implement the source and destination parts of an event-driven mechanism.

4. CONCLUSIONS

Software development in Web environment is different from that in homogenous environment. Common software engineering methodologies can still be applied, but must be modified to accommodate the nature of Web pages as the unit of software component. This is especially true in the detailed design phase.

It is also possible to migrate some types of non-Web applications to run in Web environment, but this may also need major redesign. Another obstacle in the migration is the inability of Web to work with user dialogs that involve advanced event-driven mechanism.

ACKNOWLEDGEMENTS

The author wishes to thank Dr. Achmad Djunaedi, the Head of the Computer Center, Gadjah Mada Univiersity, for his permission to use the equipments in the Computer Center during the research.

REFERENCES

_____. 1996. The Common Gateway Interface Specification. National Centre for Supercomputing Application (NCSA), University of Illinois at Urbana-Campaign. Web page, URL : http://hoohoo.ncsa.uiuc.edu/cgi/.

Booch, G. 1991. Object Oriented Design with Applications. Addison-Wesley.

Cornell, G. and Horstmann, C.S. 1996. Core Java. SunSoft Press., CA.

Hughes, D. 1997. Mini SQL 2.0 Beta Documentation. Hughes Corporation. Mini SQL Web page, URL : http://hughes.com.au/.

Kernighan, B. and Ritchie, D. 1988. The C Programming Language. Second Edition. Prentice-Hall, NJ.

Lerdorf, R. 1997. *PHP/FI version 2.0.* Web page, URL : http://www.vex.net/php/.

McGuirk, P. 1996. *The HyperText Markup Language for Netscape Navigator 3.0.* Web document, URL : http://www.slip.net/~mcguirk/html-spec.html.

Pacheco, X. and Teixeira, S. 1995. *Delphi Developer's Guide.* Sams Publishing, Indiana.

Pressman, R. 1992. *Software Engineering : A Practitioner's Approach.* Third Edition. McGraw-Hill, Singapore.

Raggett, D., LeHors, A., and Connolly, D. 1997. *HyperText Markup Language (HTML) : Working and Background Materials.* World Wide Web Consortium (W3C).

Segal, B. 1995. *A Short History of Internet Protocols at CERN.* Web document, URL : http://wwwcn.cern.ch/pdp/ns/ben/TCPHIST.html.

Wall, L. and Schwartz, R.L. 1991. *Programming Perl.* O'Reilly & Associates, Inc., CA.