Evaluating Programmer and Graphic Designer Interaction Using the Model 1 and Model 2 Web Development Architectures

Michael Seaholm, Laura Soderlund, Sarah Murto,
Joline Morrison, and Mike Morrison
Department of Computer Science
University of Wisconsin-Eau Claire
Eau Claire, WI 54702
morrisjp@uwec.edu

Abstract

Clear communication is essential to any project, but when people from different fields work together on a project, it can be difficult to interact efficiently and effectively. This can be especially challenging with web projects that require combining HTML and server-side programming commands into one program. This paper describes a case study exploring the process of creating an application using different web development architectures to find if a specific architecture supports more effective interactions between the programmer and web designer. The study found that interactions went most smoothly using an architecture whereby the program code and design code is stored in separate files.

Introduction

Clear communication is essential to any project, but when two people working on a project are from different fields, it can be difficult to interact efficiently and effectively and finish the project smoothly. This problem arises for applications involving web pages that include server-side program code when a web designer, who deals with the layout code, and a programmer, who deals with server side code, need to combine their work. These two jobs use different skill sets, programming languages, and talents that need to mesh together successfully.

Different Web architecture frameworks exist to facilitate development of Web sites that contain server-side program commands. To explore which type of architecture works best, a programmer and a web designer developed a program using three different frameworks. They first implemented the program with Java Server Pages (JSPs), which follows the Model 1 architecture. They then replicated the implementation using Model 2 architecture, also known as Model View Controller (MVC), and implemented the program in both Struts and ASP.NET MVC. An independent observer tracked their interactions and noted their progress as they worked. This paper summarizes what was observed through this research, and provides insight into how these different architectures and environments facilitate programmer and designer interaction.

Previous Research

As far as we are aware, no studies have been done to date that have actually observed and documented graphic designer and Web programmer interactions using different Web development architectures. One study done by Reed and Davies [2] discussed the use of various Web technologies from the perspective of a computer programmer and a designer, but did not explore different approaches with the goal of optimizing interactions between the two parties. Instead, the paper describes how programmers and designers individually apply aspects from both fields.

Many published papers describe the Model 2 architecture and its underlying specification, with the assumption that the Model 2 architecture is superior to Model 1. However, none specifically address whether Model 2 succeeds in facilitating designer and programmer interactions. A study done by Heidke et. al. [1] compared several different approaches to web design, separated into Model 1 and Model 2 architectures. Heidke acted as both the graphic designer and programmer for this web application, and therefore did not address designer/programmer interaction.

Generally, web sites and web applications conform to one of two different web architectures, Model 1 or Model 2. Implementations of the Model 1 architecture are characterized by a sharing of the design code that defines the view with the server-side code that makes up the business logic of the application. The two types of code are expressed in each page without any significant form of separation, and as a result, it may be difficult to understand, create, and modify the code. In contrast, the Model 2

architecture stipulates that the design code and the business logic is kept separate in different files or other structures. The Model-View-Controller (MVC) design pattern adheres to this standard and sees use in the development of web applications [1]. Since the model and the view are separate in this setup, a structure known as a controller is typically used to route information between the two. The Model 2 architecture is widely accepted as an easier way to develop web sites and applications owing to this separation of design and server-side code.

Research Methodology

The research methodology is a combination of case study research and action research [3, 4]. Action research is the systematic, reflective study of one's actions and the effects of these actions in a workplace context [3]. Using this approach, the designer and programmer recorded their own interpretations of the development process along with their views of what web architecture best facilitates programmer and web designer. For the case study portion of the research, the designer and programmer were overseen by a third outside observer, who recorded the interactions between the programmer and graphic designer.

We selected these methodologies because they are well suited to a preliminary investigation. A lab experiment was eliminated due to difficulties in coming up with additional pairs of programmers and graphic designers.

Overview of the Study

The following paragraphs describe the program that the study participants created and provides details about the research process.

The Program

The programmer and designer implemented a grade book program named GradeLink. The goal of GradeLink is to allow faculty members to distribute grades to students easily and securely. The faculty member creates the grade book in Excel and places it in a specific network folder location. Students who wish to view their grades can log into the GradeLink website using their network account and see their current classes displayed. They can select a class and choose a category of grades to view. (These categories depend on how the faculty member organizes the grade book using Excel workbook pages.) Once the student selects a category, he or she can view their grades that fall under that category as well as individual comments related to each grade.

Research Process

Meetings between the programmer and web designer typically occurred twice a week. At these meetings, their individual work would be combined and evaluated. The observer would attend all group meetings, as well as receive any e-mails that were sent between the programmer and designer. She would take notes on interactions and comments that were made. She also conducted interviews with the programmer and designer on a monthly basis depending on how much progress was made on the project.

Observations considered important, and questions used in interviews are based off basic anthropological principles and advice provided by Robert Yin's book *Case Study Research: Designs and Methods* [4]. Yin suggests that an observer keeps track of documentation, performs interviews, and directly observes the subject of research. When the observer comes to a conclusion, it is "likely to become much more convincing and accurate if it based on several sources of information" (p. 97). As also suggested by Yin, all of the notes were kept in a 'database' of information on the observer's computer, and were not released to the group until the observation period was over.

- What do you feel worked very well since the last interview?
- What do you feel has not been working well since the last interview?
- Has there been a situation where compromise has been made and you feel another solution could have been more efficient?
- Are there any feelings you have been having/thought about how things are going that you feel have not been expressed during collaborative meetings?
- For upcoming meetings, what are you planning to do to try and make things progress smoothly?

Figure 1: Interview Questions

The observer tried to record as many details as possible about discussion between the programmer and web designer. This would range from what questions they asked each other, to tones of voice that indicated confusion or frustration. Progress on the project was also noted for general timeline purposes. The notes would then be transferred into the database to be read over when it was time to look at the research as a whole.

Interviews would consist of the questions in the sidebar, as well as additional clarification questions, or questions more specific to project progress. Interviews were performed privately so that all thoughts could be expressed. Questions were left as open-ended as possible so that the respondent did not feel something they wanted to say was not related to the question. Interview transcripts would then be added to the database. The database also included copies of all the e-mails sent between the programmer and designer, rough drafts of the website, polls taken by students to choose a layout and other paperwork that was used in the project.

Results and Observations

Model 1 (JSPs)

The first implementation that was completed for this research project was based on the Model 1 architecture. This program made use of Java Server Pages (JSPs) for the basis of the application. As is typical of the Model 1 architecture, this project consisted primarily of two components: the view, as represented by HTML code, and the model, or Java code embedded directly into the HTML. Similar to a standard web page, formatting information for the view can be specified using a cascading style sheet (CSS). The difference, however, is that the embedded Java code can be used to provide server-side functionality that cannot be expressed with plain HTML.

Since the Java code in JSP pages is interleaved with the web design (HTML) code, it can be difficult for the web designer to know where he or she can insert code. In this study, the designer said, "Struts makes it easy because my code is separate, but… it wasn't hard to search through [the programmer's] code [in JSP]." However, the designer's ability to parse through the programmer's code was limited because the programmer interleaved the different types of code together most of the time. The designer still had to look through the code to edit things, but did not work with any large code transfers. The programmer did not seem to mind this however. He stated, "I can read HTML well, [although] it's not helpful for [the designer] if some part of the code needs to be moved, but it's easy for me. Better than working through different layers, when [in JSPs] it is more integrated."

When revising the JSP implementation's layout, some code for a table needed to be placed inside a Java scriptlet. The designer needed the programmer to direct where the code could go, and eventually gave up and let the programmer fix it. The designer was confused by what the scriptlet was doing, and found the idea of coding inside the scriptlet slightly overwhelming. However, no problems were encountered when adjusting the layout when the designer and programmer code did not overlap extensively.

The designer did show hesitation towards dealing with Java programming code in general. She commented when working one day that "Java confuses me, so I'll just back away a little bit." When asked what was not going well during an interview, the answer was "Java….Sometimes I don't know what I'm doing or why it's not working." Also, the designer noted that "I'm not a programmer. It's also hard to get a grip on the concept [of some of the programming and programming environments] since they're all new concepts to me."

The designer showed clear hesitancy towards the new technology and unfamiliar ideas. Although these concepts can be learned, the hesitancy can slow down a project and cause it to go less smoothly with an inexperienced designer. It should be kept in mind that JSPs were the first implementation of GradeLink, and therefore the first time working in the programming environment and being exposed to server side programming. Therefore,

these hesitancies in the later implementations may have been less obvious due to exposure.

Model 2 (Struts)

The next implementation that the programmer and the designer worked on was one of two that would follow the Model 2 architecture. Like the previous implementation, the Struts application made use of JSPs to link information together. However, the embedded Java code that was persistent throughout the JSPs in the first program was moved into a number of files elsewhere in the Struts project structure. A controller object was used to map requests made by the view to operations defined in the model. In this way, the application was identical in function to the first implementation while keeping its internal components separate.

The Struts implementation went very smoothly. The programmer simply put the designer's files where they needed to be and the system worked. Although the designer ended up hardly touching the Struts code initially, it was mentioned that "Struts makes it easy because my code is separate, but it probably won't be too much of a benefit because it wasn't hard to search through [the programmer's] code [for JSPs]". When it was necessary to go back into the code to implement the final design pattern, things seemed to go smoothly as well. Towards the end of the study, the designer mentioned a preference to work in Struts first, and then transfer the work to the other environments.

Since the programmer could reuse the code from the JSP implementation, Struts went very smoothly for the programmer as well. He stated, "the good thing about Struts is the view is separate." There was an initial struggle with taking the HTML out of the JSP code, but otherwise it went smoothly. Struts was liked overall by both parties with few negative comments towards it, but it should be noted that they were more used to the Eclipse Java IDE that was used for developing the Struts and JSP implementations than Visual Studio 2008 for ASP.NET MVC, which is described in the next section.

Model 2 (ASP.NET MVC)

The final iteration of the grade book viewing system is also derived from the Model 2 architecture. It marks a deviation from the previous implementations in that it was developed on a non-Java platform, ASP .NET MVC application. It was created in the Visual Studio 2008 environment, and written in HTML and C#. Rather than JSPs, the ASP .NET MVC implementation made use of web forms with an .aspx extension for the design code. Additionally, where the model and controller in the Struts application were stored in Java source files, ASP .NET MVC places the business logic of the application into sections of the controller in a C# source file. The organizational flow of the ASP .NET MVC implementation follows the Model 2 architecture while providing a different development environment and programming language for the developer.

ASP.NET MVC did not go as smoothly as Struts, partially because it was a new environment for both the programmer and designer. The programmer had previously used Struts and JSPs, but had not used ASP.NET MVC, and the designer had not seen it before. Since it was a new environment, this part of the project was interesting because it explored how well MVC supports interactions when the programmer and designer work with an unfamiliar environment.

There were several aspects of ASP.NET MVC code that were sources of frustration. When the layout was added, it looked significantly different from the previous implementations, although no code was changed. Most noticeably, windows were shifted in various directions and separated from one another, and at first it was not clear why this was happening. The programmer also seemed to struggle slightly with learning C# and the Visual Studio 2008 IDE.

Although there were many difficulties involved with debugging code for ASP.NET MVC, feelings towards it weren't too harsh. Though both the designer and programmer indicated that debugging was difficult, both thought it was easy to code once they were used to the environment. Many of the negative comments towards the language were attributed to not being used to it: the programmer explicitly stated that part of the dislike was due to it being different from the Eclipse development environment.

Discussion

Having an appropriate architecture can aid the progress of a project, but it will not cause a project to go smoothly on its own. Several characteristics of the individual programmer and designer helped aid the progress in different architectures. For instance, the programmer knew HTML and could manipulate the HTML design code easily. Both parties were used to working in a Java environment, although the designer was less experienced than the programmer. When the designer did get used to the environment, her comfort level with editing someone else's code rose. As the study drew to a close, the comfort level with ASP.NET MVC was also beginning to improve as well.

Communication was generally successful in all architectures: throughout interviews, both parties would say that communication was going well. Both parties also stated in their final interviews that meeting, discussing their problems, and looking at the code together helped immensely with the progress of the project. Communication may also have been aided by the programmer and designer getting along well and not being afraid to state their thoughts.

Although this project was completed smoothly in all architectures, bumps in the road came up occasionally. Most of the difficulties arose from troubleshooting and general problem solving that comes with software design. This was most noted in the frustration expressed towards ASP.NET MVC, although there were other issues that arose. Some frustration was also expressed towards not knowing exactly what design results were

wanted for the final product, but when these situations occurred, the programmer or designer would state their problem to the group and a decision would be made.

Conclusions

During final interviews, both the programmer and designer stated that they preferred the Model 2 architecture using Struts. The programmer was firm in his position, saying "Struts is ideal. JSPs are clumped together and can be inconvenient for group work, but Model 1 is still OK in general. ASP.NET MVC was difficult, so the goals of the project weren't met." The designer was not as emphatic, stating "I still like JSPs or Struts [rather than ASP.NET MVC], because I'm more used to it." Notably, both stated that Struts is easier because of experience.

It is interesting to note that the designer did not seem to have a strong preference towards the Model 2 architecture. The Model 2 architecture was developed to separate the program code from the design code, so this was surprising. The project seemed to be most impacted by familiarity with the programming environment and good communication between the parties involved rather than using a particular architecture.

Future research might replicate the study using experienced programmers and designers to see if experience level changes interactions and preferences. It would also be interesting to see how smoothly a project would go when both the designer and programmer had less or no experience with the other's field. Finally, research could involve conducting a similar case study, but having each implementation of the program be distinctly different so code could not be reused.

References

[1] Heidke, N., Morrison, J., and Morrison, M., "Assessing the Effectiveness of the Model View Controller Architecture for Creating Web Applications." Midwest Instruction and Computing Symposium, Rapid City, SD, April, 2008, http://mics.sdsmt.edu/proceedings/Paper/mics2009_submission_55.pdf.

[2] Reed, David and Joel Davies. "The Convergence of Computer Science and Graphic Design." *Consortium for Computing Sciences in Colleges* (2005) 179-187. Web. 9 Sep 2009.

[3] Riel, M. (2007) Understanding Action Research, Center for Collaborative Action Research. Pepperdine University. 9 Sep 2009. http://cadres.pepperdine.edu/ccar/define.html

[4] Yin, Robert K. Case Study Research. Revised. Newbury Park: Sage Publications, 1989. Print.