Navigating the Framework Jungle for Teaching Web Application Development

Joseph T. Chao
Bowling Green State University
Bowling Green, Ohio, USA

Kevin R. Parker
Idaho State University
Pocatello, Idaho, USA

Bill Davey
RMIT University
Melbourne, Australia

jchao@bgsu.edu parkerkr@isu.edu Bill.Davey@RMIT.edu.au

Abstract

Studies indicate that information systems and computer science programs should place more emphasis on software design topics. Because little significant software in a commercial environment is developed using only programming skills, students without exposure to design patterns and frameworks will be ill-prepared for the workforce. This paper investigates whether PHP-based web development courses should use PHP frameworks to guide program development and how to select an appropriate framework. The paper begins with a literature review of the concept of design patterns, particularly the Model-View-Controller (MVC) pattern. We review studies that argue that design patterns and frameworks are essential pedagogical tools. Finally, we report on our work to select an appropriate framework for incorporation into web development courses as well as the process by which some of the major PHP MVC frameworks were assessed to determine which is best-suited for use in an academic environment.

Keywords: Web application development, design patterns, Model-View-Controller, software design, PHP MVC framework, software selection.

Introduction

Software design concepts are often underemphasized in information systems and computer science curricula. "Textbooks frequently take the approach of demonstrating how to use a particular library or toolkit, spending little time discussing the structure of the program or architectural patterns like MVC" (Hanson & Fossum, 2005, p. 120). The IS 2010 Curriculum Guidelines (Topi et al., 2010) state that graduates of undergraduate IS programs should be experts in high level design of enterprise architecture. "Although the knowledge and skills that IS graduates need have recently moved significantly in the direction toward higher levels of abstraction, individual skills related to design and implementation are still essential for IS graduates" (Topi et al., 2010, p. 20). The Computer Science Curriculum (Cassel, 2008) guideline includes eight core hours dedicated to design topics such as design

patterns, software architecture, structured design, and object-oriented design, but in many programs it seems that several of those topics are not adequately emphasized. Students are often so focused on learning programming language syntax that they fail to see the big picture of applying that language to solving real problems (Hundley, 2008).

The purpose of this paper is to review the need to use frameworks in the classroom and if so, how to select the most appropriate framework. We begin by examining the extant literature confirming that educational gains can be made through using frameworks to teach software design concepts. We selected the web application development course as a viable candidate for the inclusion of frameworks because such courses are often PHP based, and there is a variety of PHP frameworks from which to choose. We selected several viable PHP frameworks for consideration, and then applied a structured selection process to determine which framework would best serve our purposes. Hence, the scenario for this discussion requires that:

- The curriculum includes a course (or courses) in web development.
- The decision has been made to use the PHP programming language, generally considered the most commonly used open-source server-side scripting language.

Overview of Design Patterns, MVC, and Frameworks

While the focus of this paper is on frameworks, frameworks often contain implementations of many cooperating design patterns. Thus, design patterns will serve as the starting point for this overview.

Design patterns are a fairly recent concept in software development. They were first proposed in an OOPSLA-87 Panel Session by Cunningham and Beck (1988), who in turn attribute the idea to the architectural work of Alexander (1977). The concept went relatively unnoticed until the seminal piece by Gamma, Helm, Johnson, and Vlissides (1993), in which they formalize the concept of design patterns and propose a design pattern template.

A design pattern is a reusable solution that designers can apply, with possible modifications, to an application. It presents the interaction of data and methods in multiple classes, describes the implementation of classes and features needed for the interaction, and offers insight to problems that may arise in the application.

Over the last two decades software professionals have embraced patterns as a means of recording and sharing expert knowledge for building software (Wallingford, 2002). Experience often makes the difference between being a poor and good designer. It seems sensible that students should be exposed to this approach to design.

There is a variety of design patterns in widespread use, including the Model-View-Controller (MVC) pattern, Singleton, Application Controller, Builder, Front Controller, Iterator, Observer, Registry, and State (Paikens &Arnicans, 2008). MVC is arguably the best known, most famous design pattern (Turner & Bedell, 2002). It was originally designed for applications that provide multiple views of the same data, but now it is virtually the central feature of modern interactive applications (Morse and Anderson, 2004).

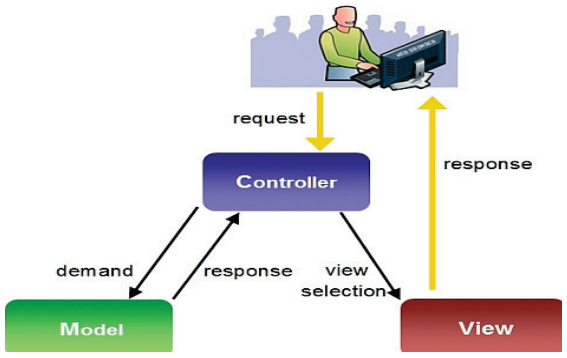

Figure 1: MVC Patterns.

The key idea behind the MVC pattern was based on the "input-processing-output" view of systems with the goal of separating user interfaces from the underlying data represented by the user interface. Thus, the MVC pattern, as shown in Figure 1, separates the three main components of an application: models, views, and controllers.

Models manage application data, views present the user interface, and controllers handle events for views. Separating the internal data models from the interface views that set or display the data allows models to be represented by a variety of views. Further, the model's algorithmic logic is distinct from the management of the view, thus leading to a simpler program design.

The MVC pattern has proven to be very useful in industry and also can be effectively used in student projects using either traditional or object-oriented methodologies (Zant, 2006). MVC frameworks exist for many languages and systems including Java, Python, PHP, Microsoft ASP, and Apple's Cocoa application environment.

The MVC architecture pattern is often implemented with a software framework such as WebSphere, Struts, or Zend. A framework defines a family of related applications, and contains elements that are common to those applications (Tao, 2002). Frameworks provide structure by enforcing naming conventions (directories, files) and rules for constructing a system. They also provide components that aid in the construction of a system. Application developers extend the framework to build custom applications. Thus, a framework is basically a development environment in which programmers can develop applications of all types much easier and faster, including web applications. A framework may consist of source code libraries, utilities, plug-ins, development models, and a wide range of tools, the purpose of which is to accelerate the development pace of an application.

There is a core of recurring themes in most frameworks (Caspersen & Christensen, 2008):

- The framework delivers application behavior at a high level of abstraction.
- A framework provides functionality within a well-defined domain, i.e., frameworks address specific domains like graphical user interfaces, gaming, insurance, telecommunication, etc.
- A framework defines the interaction patterns between a set of well-defined components/objects. To use the framework the developers are required to understand these interaction patterns and must program in accordance with them.
- A framework is flexible so that it can be tailored to a concrete context, provided that context lies within the domain of the framework.
- A framework provides reuse of code as well as reuse of design.

Figure 2 depicts an example project, built on a PHP framework, in which the software flow is executed in the following way:

- The user makes a request for a page from the server. This is either a standard HTTP call or an AJAX HTTP call.
- The request is intercepted by index.php, which bootstraps the framework.
- After the framework has been bootstrapped it executes the index action of the controller. This action creates an instance of the module data model for the module name provided as part of the URL for the request.
- The module data model loads the module XML file and in-turn creates an instance of the text data model. The text data model is responsible for loading the text for the module from all the various sources.
- After the module data model is created, the controller then tells it to render the page requested. The module finds the page entry in the parsed XML file, reads out the contents, and then renders the page. The actual page rendering generally requires translating paths

for the resources on the page (images, flash movies, and the like) and then replacing text placeholders with the correct text.

- After the rendered contents are retrieved, they are passed to either an AJAX response (in the case of an AJAX request) or passed to the view script that outputs the page contents to the user.

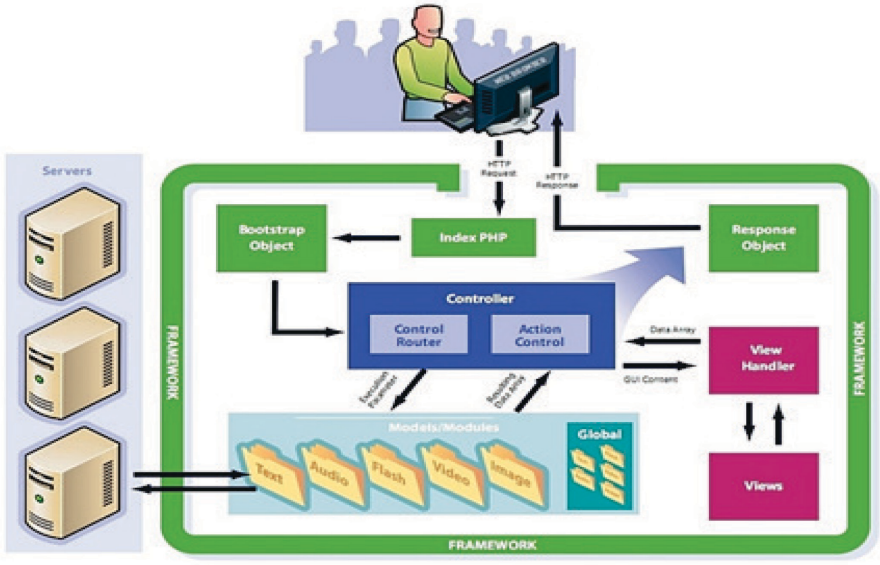

Figure 2: Example MVC Framework.

Benefits of Design Patterns and Frameworks

The motivation for using design patterns and frameworks is explained by Ali, Bolinger, Herold, Lynch, Ramanathan and Ramnath (2011), Gamma et al. (1993), Vuksanovic and Sudarevic (2011), and Wick (2001). Design patterns facilitate the reuse of successful architectures, because expressing proven techniques as design patterns makes them more readily accessible to other developers.

Most commercial software is developed using a framework by extending and customizing the default, generic functionality that it provides. Thus, frameworks not only help in understanding and constructing real world applications, but also provide additional reusability of both design and implementation. Frameworks offer numerous technical and organizational advantages over classical development methods, such as faster development and cleaner application structure.

The use of frameworks also reduces time spent on code maintenance and future development in many ways:

- Frameworks provide numerous libraries and helpers, making it possible for developers to achieve comparable functionality with less code, so the software is more easily maintainable.
- Software unit testing, defined as a process that includes the performance of test planning, the acquisition of a test set, and the measurement of a test unit against its requirements (IEEE, 1987), is facilitated by the framework rather than performed manually.

- Those tasked with software maintenance will require significantly less time deciphering existing code developed via a framework, thanks to the MVC pattern and other coding conventions.
- Future improvements, such as protection measures against future hacking attack techniques and yet unknown vulnerabilities, may be addressed by future releases of the framework, and may only require upgrading the framework to a newer version.

Motivation for Using Frameworks in Teaching

Christensen and Casperson strongly advocate frameworks in teaching, and provide many convincing arguments for their use (Caspersen & Christensen, 2008; Christensen, 2004; Christensen & Caspersen, 2002). The programming context faced by most programmers today is radically different from the one that existed a decade and a half ago. This means that we need to teach new skills to our students in addition to the old ones. Being a successful developer is no longer a question of being a good programmer, but just as much a question of understanding complex interaction patterns in frameworks and being able to design in accordance with their guidelines. Developing any realistic software requires the ability to identify proper software frameworks to reuse as well as the skill to write the specialized code for the task at hand.

While the main motivation for teaching frameworks focuses on the obvious need to teach students about the techniques that govern modern software development, there are additional pedagogical aspects that make frameworks a worthwhile topic.

- Student motivation: If students must program everything from scratch, then the workload and complexity rule out developing software that in any respect compares to the sophisticated and appealing programs that they are used to form, for example, the Windows platform. However, a framework used in the classroom defines the skeleton of an application that can be customized by an application developer and can provide the "bells and whistles" that make the effort invested by the student look more appealing or "professional". The "return on investment" is simply greater for the student.
- Setting an example: A well-designed framework can serve as an exemplary use of design patterns by making it clear that design patterns work together, and that patterns really define roles.
- Gentler learning curve: Simple frameworks can illustrate basic concepts in framework theory and practice in a gentle way as stepping stones for learning more complex GUI frameworks. It is important that students learn the basic concepts and techniques relating to frameworks.
- Developers should reuse: Students must learn that software development entails more than producing code, but also involves locating and incorporating reusable assets. Frameworks illustrate many design patterns and reinforce the concept that patterns express roles in a collaboration pattern rather than objects that are to be copied into a design. Frameworks exemplify the reuse of design as well as code, whereas design patterns demonstrate only design reuse.
- Workforce ready: Frameworks have already made an impact on how industrial software is developed in a cost efficient and reliable way. Graduates should be familiar with the basic concepts and techniques relating to frameworks. Further, students are better prepared for the workforce when exposed to commercial tools.
- Object concepts. Frameworks utilize best-practices of object-oriented programming. Requiring students to read and understand parts of frameworks will provide them with important insights in how to structure complex systems in a master-apprentice fashion.
- Responsibility-driven design: An MVC framework is an excellent example of how responsibility-driven design influences the identification of classes and the assignment of

responsibilities to those classes. While many students have an initial expectation that the GUI should be integrated with the data since they are so closely related, an MVC framework provides an intuitive and concrete example of how the separation of manipulation and display can result in a powerful and flexible software design (Wick, 2001).

Selecting a PHP Framework for the Classroom

Recall that the purpose of this paper is to determine whether to use frameworks to teach web application development courses, and if so, how to select the most appropriate framework. As seen above, the literature presents many convincing arguments in favor of teaching design concepts via frameworks. However, once it has been decided to use frameworks, educators still face the challenge of selecting the best framework for the class. Our web application development course uses the PHP programming language, so our focus can be narrowed to PHP frameworks.

There are many PHP frameworks available, and making the right choice can be daunting. Many PHP developers prefer to adopt a PHP framework that has a large user base and community support, while others may focus on useful features. Some developers simply consider frameworks that are popular, trendy, or widely used.

For those developers concerned with features, various framework comparisons can be found online, such as the PHP Frameworks Comparison (socialcompare.com/ en/comparison/php-frameworks-comparison) and PHP Frameworks (www.phpframeworks.com/). These provide feature-by-feature comparisons of several frameworks. PHP MVC Frameworks (www.phpwact.org/php/mvc_frameworks) also provides an extensive annotated list of PHP frameworks.

To narrow down the list for comparison, SHELDMANDU (sheldmandu.com/php/php-mvc-frameworks/choosing-the-best-php-mvc-framework-part-1) included the following best known frameworks: Zend, Yii, CodeIgniter, Solar, Doo, Symfony, CakePHP, ezComponents, Seagull, Ash.MVC, Qee, and Think PHP. SHELDMANDU then narrowed his choices down to Yii, Kohana, Zend, and CodeIgniter based on the aspects of performance, cohesiveness, comprehensiveness, feel, documentation, community, IDE support, time around, and interoperability.

The blog page called The Best PHP Framework (davidjconnelly.wordpress.com/2011/07/03/the-best-php-framework-of-2011/) provides arguments for his top five best PHP frameworks of 2011 as 1. CodeIgniter, 2. Yii, 3. Zend, 4. FuelPHP, and 5. CakePHP.

There is a variety of ways to determine the more popular or trendy frameworks. One approach is to aggregate the number of searches for a particular software tool over a specific time span. This approach is commonly used to develop various usage indexes such as the TIOBE Programming Community Index (www.tiobe.com/index.php/content/paperinfo/tpci/index.html), which is a list of programming languages ordered by the frequency of web searches using the name of the language as the keyword. A similar ranking of the most used PHP frameworks based on Google searches worldwide for each PHP framework name over a one month time frame is presented by PHP Developer (www.php-developer.org/most-used-php-framework-the-popular-top-7-list-in-year-2011/). Figure 3 shows the results of their ranking, with Zend Framework taking the top honors.

Figure 3: PHP Frameworks Search Frequency.

Another approach for determining the most widely used framework is to poll developers. One online poll (css.dzone.com/articles/php-frameworks-poll-results), the results of which are shown in Figure 4, had more than one thousand respondents. Again, Zend Framework takes top honors, but by a much closer margin.

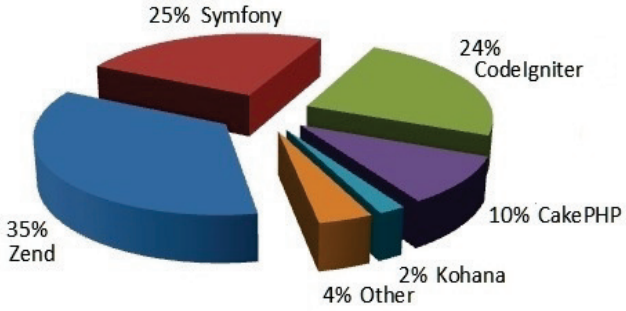

Figure 4: PHP Framework Poll Results.

Figure 5 shows the use of Google Trends (www.google.com/trends/) to show and compare the popularity and search volume of five major MVC PHP frameworks. Zend again tops the contenders.

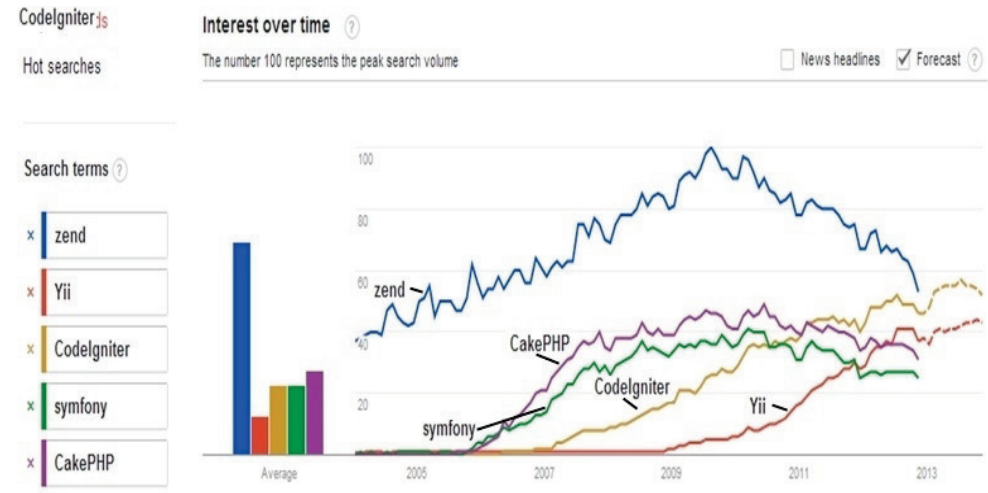

Figure 5: Google Trend Statistics for PHP Frameworks.

Selection Process

While the above approaches may be useful for professional developers, for teaching purposes a specialized selection approach may be required. Parker (2010) proposed a systematic approach for a general selection process for software tools for IS and IT curricula:

1. Compile a list of criteria.
2. Weight each of the criteria.
3. Determine a list of candidate tools.
4. Rate each software tool.
5. Calculate a weighted score.

Step 1: Compile a list of criteria

Parker (2010) also suggested a set of criteria for use in the selection of software tools for IS and IT curricula. The criteria were derived by assessing factors considered in the process of choosing software tools for various teaching purposes as discussed in well over fifty papers that reported on the selection of software for IT and IT curricula (Parker, 2010). Each of the criteria in Table 1

has been used in one or more previous studies that evaluate software tools, and so it was decided to use those criteria to satisfy Step 1.

Table 1: Outcome of Step 1 -Criteria for Software Selection.

Software Cost • Reasonable financial cost • Availability of affordable software version
Acceptance in Academia • Academic acceptance • Availability of textbooks
Industry Penetration • Maturity • Industry acceptance • Marketability of graduates
Software Characteristics • System requirements • Operating system dependence • Open source (versus proprietary) • Feature set
Pedagogical Features • Ease of learning basic concepts • Easy-to-use environment • Advanced features for subsequent courses • Collaboration support
Course Methodology and Software Paradigm • Support for teaching approach • Software paradigm
Support and Required Training • Availability of documentation and support • Training required for instructors and support staff

A complete literature review and justification for each of the criterion can be found in Parker (2010), but a brief explanation of each follows.

Software Cost

The software cost category includes two criterion, financial cost and availability of an academic version. Financial cost refers to the price to acquire the software tool. The availability of an academic version makes the tool more affordable.

Acceptance in Academia

Acceptance in academia refers to the degree to which the software tool under consideration has been embraced by academia, as well as the availability of texts.

Industry Penetration

Industry penetration refers to the degree to which the software tool under consideration has been embraced by the professional community, and encompasses the maturity of the software, its use across industry, and by extension the need for graduates with experience using the software.

Software Characteristics

The software characteristics category encompasses various software traits like system requirements, operating system limitations, the source availability model, and software feature set. While

the first three traits are self-explanatory, feature set refers to the fact that some software tools are available in an educational version with limited features.

Pedagogical Features

Pedagogical features include a language's learning curve, ease-of-use features associated with the software tool, advanced features that could support subsequent courses, and whether the software tool supports collaboration.

Course Methodology and Software Paradigm

The course methodology and software paradigm category focuses on how the software will be used in the educational environment, including the teaching approach that is preferred as well as the software paradigm.

Support and Required Training

Support and required training considers the level of and need for documentation, training, and support for both instructors and students.

Step 2: Weight each of the criteria

Each evaluator was asked to weight, specific to the department's needs, the value of the importance for each criterion. They were asked to assign weights ranging from zero (do not care) to ten (extremely important). The evaluators then convened in a meeting in which they discussed the ratings and reached a consensus on the assignment of weights. The following provides a brief recap of factors that were considered.

There are seven criteria to consider:

- Pedagogical Features
- Industry Penetration
- Support and Required Training
- Software Cost
- Software Characteristics
- Course Methodology/Software Paradigm
- Acceptance in Academia

The evaluators weighted Pedagogical Features and Industry Penetration as the most critical factors in considering framework adoption for the classroom. Pedagogical features, particularly ease of learning, can not only motivate students but also can allow them get up to speed quickly and allow enough time in a semester for in-depth understanding (Parker, 2010). Ease of learning concepts refers to the learning curve associated with framework complexity for those new to frameworks (Kao, Tousignant, & Wiebe, 2000). Industry penetration refers to the use of the software tool in business and industry. It can be assessed based on current and projected usage, as well as the number of current and projected positions. Thus, industry penetration can often be translated into marketable skills that influence future employability of graduates. Adopting industry-based software tools and development technologies can help to align student experiences in the classroom with workplace practices (Lancor & Katha, 2013).

The evaluators also considered Support and Training to be another important criterion because there must be ample resources for the instructor and students to learn how to use various aspects of the framework. This criterion looks at the level of and need for documentation, training, and support for both instructors and students. This criterion should also take into account the availability of documentation and online resources (Beise, 2006).

Although the importance of each of the remaining criteria was assessed and weighted, the characteristics of PHP frameworks make those remaining criteria less relevant.

- Software cost: each of the PHP web application frameworks that are likely candidates for use in the class is open source, meaning there is generally no software cost.
- Software characteristics: all of the PHP web application frameworks that are likely candidates for use in the class have similar features and are available on all platforms that run PHP, which lessens the impact of this criterion.
- Course Methodology/Software Paradigm: all of the PHP web application frameworks that are likely candidates implement the MVC design pattern, so this criterion has little impact.
- Acceptance in Academia: it is the perception of the evaluators that frameworks are not widely taught in IT curricula, so this criterion is of reduced consequence.

The final weights for the criteria are shown in Table 2.

Table 2: Outcome of Step 2 – Weight each criterion.

Criteria	Weight
Pedagogical Features	10
Industry Penetration	8
Support and Required Training	7
Software Cost	5
Software Characteristics	5
Course Methodology/Software Paradigm	3
Acceptance in Academia	3

Note that the weightings assigned to each criterion are dependent on the needs of whatever individuals are involved in the selection process. The evaluators placed more weight on a gentle learning curve, but that may not be the case with everyone who is assessing frameworks for their own teaching purposes.

Step 3: List Candidate Tools

Based on observed trends and on the above criteria, particularly ease of learning and industry popularity, the frameworks shown in Table 3 were selected for comparison. Zend is indisputably the most widely used framework. CodeIgniter, Symfony, and Yii are similarly popular, although both CodeIgniter and Yii have been consistently increasing in popularity. Thus, the widespread usage criterion led to the selection of Zend, CodeIgniter, Symfony, and Yii for the study. Ease of learning led to the inclusion of CakePHP due to its reputation for having a much gentler learning curve.

Table 3: Outcome of Step 3 – List Candidate Tools.

Software Tool
Zend
CodeIgniter
Yii
Symfony
CakePHP

Step 4: Rate each software tool

Given the criteria established in Step 1, we conducted an extensive literature review, visited framework websites, and read various related discussion boards and blogs. The keywords MVC, frameworks, Zend, CakePHP, CodeIgnitor, Yii, and Symfony were submitted to several search engines and relevant sources inspected by at least two of the team. This was done several times over the period August 2011 to December 2012. As a result of this data collection it was possible to isolate critical characteristics for each alternative. These characteristics, arising from examining assessments by both proponents and opponents of each alternative, were then used to evaluate each of the criteria to be considered. In addition, one of the authors downloaded, installed, and used the frameworks for several web development projects over the last four years. The professor also had students download, install, and use various frameworks to assess ease of use. Based on the findings, the PHP frameworks were rated for each of the criteria using a Likert Scale 1 (worst) to 5 (best) in Table 4.

CakePHP is known for its ease of use and Zend for its complexity, with CodeIgniter and Yii rated as moderately easy to use and moderately complex (Porebski, Przystalski, & Nowak, 2011). Zend is no doubt the most widely used framework, while CakePHP is on the opposite end of the spectrum. With regard to support and documentation, all four frameworks fared well, with CakePHP having the best documentation. The frameworks were given almost identical ratings for the final four criteria, because there are few differences in the frameworks with regard to these factors.

Table 4: Outcome of Step 4 – Rate each software tool.

Criteria	Zend	CodeIgniter	Yii	CakePHP	Symfony
Pedagogical Features	2	3	4	5	3
Industry Penetration	5	4	4	1	4
Support and Required Training	3	4	4	5	3
Software Cost	5	5	5	5	5
Software Characteristics	4	4	4	3	4
Course Methodology/Software Paradigm	5	5	5	5	5
Acceptance in Academia	1	1	1	1	1

Step 5: Calculate a weighted score

This step is simply the result of multiplying the assigned ratings by the weights for each criterion and then summing the weighted scores for each framework. The results are shown in Table 5.

Note that Yii received the highest weighted score, and thus for this particular scenario it was determined that Yii would be the framework to be used in the web development course. This is not to say that Yii is the best PHP framework available, simply that it is the tool best suited for use in our class given our particular needs.

Table 5: Outcome of Step 5 – Calculate a weighted score.

Criteria	Weight	Zend	CodeIgniter	Yii	CakePHP	Symfony
Pedagogical Features	10	20	30	40	50	30
Industry Penetration	8	40	32	32	8	32
Support and Required Training	7	21	28	28	35	21
Software Cost	5	25	25	25	25	25
Software Characteristics	5	20	20	20	15	20
Course Methodology/Software Paradigm	3	15	15	15	15	15
Acceptance in Academia	3	3	3	3	3	3
Weighted Score:		**144**	**153**	**163**	**151**	**146**

Conclusion

The demand for web application developers is on the upswing, and with a variety of PHP frameworks available for web application development, it seems prudent to investigate their incorporation into web development courses. Several studies have made the case that current curricula neglect program structure and design patterns like MVC. One states quite bluntly that "young people begin designing Web pages without being aware of even the simplest principles of software design" (Paikens &Arnicans, 2008, p. 53).

Upon concluding that incorporating frameworks in a web development course is prudent, it becomes necessary to select the most appropriate framework for the course or curriculum. The choice of software tools in a teaching environment should be based not only on technical attributes but more importantly on student needs such as use in industry and the ease of learning. The method detailed in Parker (2010) was designed for the specific purpose of selecting software tools for use in an IS or IT curriculum. The structured approach makes software selection a more rigorous and objective process.

The first step in the process is to compile a list of selection criteria. Toward that end, Parker (2010) provides a detailed set of evaluation criteria for software selection based on extensive research. These student-centered criteria include Pedagogical Features, Industry Penetration, Support and Required Training, Software Cost, Software Characteristics, Course Methodology/Software Paradigm, Acceptance in Academia. Each of those criteria is then assigned a weighting by the individuals involved in the selection process. A set of "likely candidates" is then chosen for consideration. In this case, the candidates were Zend, CodeIgniter, Yii, Symfony, and CakePHP. Each software tool is rated by the academics involved in planning the course based on research and hands-on experiences, and then weighted scores are calculated.

The process followed here proved to be efficient and resulted in a justifiable choice. Yii received the highest weighted score among the candidates, given our individualized parameters. It bears repeating that this does not mean that Yii is the best PHP framework for every situation; rather that Yii is the preferred tool based on our given criteria. Other academics making the same selection would no doubt assign different weights and perhaps different ratings, resulting in a different framework earning the highest total weighted score, indicating that it is more appropriate for their purposes.

This paper shows how one group of teachers inspected alternatives with a view to choosing a framework. If it is accepted that frameworks are a necessary preparation for the modern information systems professional then the methods used here, rather than our specific decision, should be an aid to those making that choice for themselves.

Regardless of which framework is chosen, there are many benefits to incorporating web application development frameworks in the classroom. By introducing students to frameworks during their undergraduate education they gain not only an appreciation for design concepts like separating the user interface from the underlying data and logic, but also exposure to commercial software tools that enhances their future marketability.

References

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language.* Oxford, UK: Oxford University Press.

Ali, Z., Bolinger, J., Herold, M., Lynch, T., Ramanathan, J., & Ramnath, R. (2011). Teaching object-oriented software design within the context of software frameworks. *Proceedings of the 41st annual ASEE/IEEE Frontiers in Education Conference*, Rapid City, SD, 1-5. Retrieved November 18, 2012 from http://www.cse.ohio-state.edu/~aliz/TDCF.pdf

Beise, C. (2006). Revisiting database resource choice: A framework for DBMS course tool selection. *Proceedings of the Twelfth Americas Conference on Information Systems*, Acapulco, Mexico, 2139–2144 Retrieved November 18, 2012 from http://aisel.aisnet.org/amcis2006/266/

Caspersen, M. E., & Christensen, H. B. (2008). Frameworks in teaching. In J. Bennedsen, M.E. Caspersen, & M. Kölling (Eds.), *Reflections on the teaching of programming methods and implementations* (pp. 190-205). Heidelberg, Germany: Springer-Verlag.

Cassel, L., Clements, A., Davies, G., Guzdial, M., McCauley, R., McGettrick, A., … Weide, B. W. (2008). *Computer science curriculum 2008: An interim revision of CS 2001*. USA: Association for Computing Machinery and IEEE Computer Society. Retrieved November 18, 2012 from http://www.acm.org//education/curricula/ComputerScience2008.pdf

Christensen, H. B. (2004). Frameworks: Putting design patterns into perspective. *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, Leeds, UK, 142-145. Retrieved November 18, 2012 from http://www.daimi.au.dk/~hbc/publication/iticse2004.pdf

Christensen, H. B., & Caspersen, M. E. (2002). Frameworks in CS1 – A different way of introducing event-driven programming. *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education*, Aarhus, Denmark, 75-79. Retrieved November 18, 2012 from http://cs.au.dk/~mec/publications/conference/03--iticse2002.pdf

Cunningham, W., & Beck, K. (1988). Using a pattern language for programming. *Addendum to the Proceedings of OOPSLA'87, ACM SIGPLAN Notices, 23*, 5.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1993). Design patterns: Abstraction and reuse of object-oriented design. *Proceedings of the 7th European Conference on Object-Oriented Programming*, Kaiserslautern, Germany, 406-431. Retrieved November 18, 2012 from http://www.cs.pitt.edu/~mock/cs1530/lectures2/ecoop93-patterns.pdf

Hanson, S., & Fossum, T. V. (2005). Refactoring model-view-controller. *Journal of Computing Sciences in Colleges, 21*(1), 120-129. Retrieved November 18, 2012 from http://www.cs.uwp.edu/staff/hansen/publications/StopWatch/mvc.ccsc.pdf

Hundley, J. (2008). A review of using design patterns in CS1. *Proceedings of the 46th Annual Southeast Regional Conference on XX*, Auburn, CA, 30–33.

IEEE. (1987). IEEE standard for software unit testing. Retrieved November 18, 2012 from http://www.akitaonrails.com/files/std1008-1987.pdf

Kao, D., Tousignant, W., & Wiebe, N. (2000). A paradigm for selecting an institutional software. *Proceedings of the Information Systems Education Conference 2000*, Philadelphia, PA. Retrieved March 2, 2013 from http://proc.isecon.org/2000/207/ISECON.2000.Kao.pdf

Lancor, L., & Samyukta Katha, S. (2013). Analyzing PHP frameworks for use in a project-based software engineering course. *Proceedings of the 43rd Annual SIGCSE Technical Symposium on Computer Science Education*, Denver, CO. Retrieved March 2, 2013 from http://db.grinnell.edu/sigcse/sigcse2013/Program/viewAcceptedProposal.pdf?sessionType=paper&sessionNumber=116

Morse, S. F., & Anderson, C. L. (2004). Introducing application design and software engineering principles in introductory CS courses: Model-view-controller Java application framework. *Journal of Computing Sciences in Colleges, 20*(2), 190-201. Retrieved November 18, 2012 from http://www.tol.oulu.fi/kurssit/otekniikka/papers/MVC-framework.pdf

Paikens, A., & Arnicans, G. (2008). Use of design patterns in PHP-based web application frameworks. *Scientific Papers University of Latvia*, *Computer Science and Information Technologies.733*. 53-71. Retrieved November 27, 2012 from http://www.lu.lv/materiali/apgads/raksti/733_pp_53-71.pdf

Parker, K. R. (2010). Selecting software tools for IS/IT curricula. *Education and Information Technologies, 15*(4), 255-275.

Porebski, B., Przystalski, K., & Nowak, L. (2011). *Building PHP applications with Symfony, CakePHP, and Zend Framework*. Indianapolis, IN: Wiley Publishing.

Tao, Y. (2002). Component- vs. application-level MVC architecture. *32nd Annual Frontiers in Education*, Boston, MA, T2G7-T2G10. Retrieved November 18, 2012 from http://fie-conference.org/fie2002/papers/1278.pdf

Topi, H., Valacich, J. S., Wright, R. T., Kaiser, K., Nunamaker, J. F., Jr., Sipior, J. C. et al. (2010). IS 2010: Curriculum guidelines for undergraduate degree programs in information systems. *Communications of the Association for Information Systems, 26*, 359-428. Retrieved November 18, 2012 from http://aisel.aisnet.org/cgi/viewcontent.cgi?article=3516&context=cais

Turner, J., & Bedell, K. (2002). The Model-View-Controller design pattern: 'Model 2' JSP development. In J. Turner & K. Bedel, *Struts kick start* (Chapter 2). Indianapolis, IN: Sams Publishing. Retrieved November 18, 2012 from http://ptgmedia.pearsoncmg.com/imprint_downloads/informit/chap2_0672324725.pdf

Vuksanovic, I. P., & Sudarevic, B. (2011) Use of web application frameworks in the development of small applications. *Proceedings of the 34th International Convention on Information and Communication Technology, Electronics and Microelectronics*, Opatija, Croatia, 458-462.

Wallingford, E. (2002). Functional programming patterns and their role in instruction. *Proceedings of the International Conference on Functional Programming*, Pittsburgh, PA, 151-163. Retrieved November 18, 2012 from http://www.cs.uni.edu/~wallingf/patterns/papers/fdpe2002/fdpe2002-long.pdf

Wick, M. R. (2001). Kaleidoscope: Using design patterns in CS1. *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education*, Charlotte, NC, 258-262. Retrieved November 18, 2012 from http://ebookbrowse.com/p258-wick-kaleidoscope-using-design-patterns-in-cs1-pdf-d305384903

Zant, R. F. (2006). Model-view-controller architecture in a systems analysis and design course. *Proceedings of the Information Systems Education Conference,* Dallas, TX. Retrieved November 18, 2012 from http://proc.isecon.org/2006/3353/ISECON.2006.Zant.pdf

Biographies

Dr. **Joseph T. Chao** is an Associate Professor and the Director of the Agile Software Factory in the Department of Computer Science at Bowling Green State University. The Software Factory, which he founded in 2008 with a grant from the Agile Alliance, provides students with service-learning opportunities and real-world project experience in software engineering. Prior to entering academia, Dr. Chao gained more than seven years of industry experience working as Software Engineer, System Analyst, QA Tester, Project Manager as well as Director of Software Development. His research focus is on software engineering with special interests in agile software development, database management systems, web and mobile technologies, and object-oriented analysis and design. Dr. Chao holds an M.S. in Operations Research from Case Western Reserve University and a Ph.D. in Industrial and Systems Engineering from The Ohio State University.

Dr. **Kevin R. Parker** is a Professor and Chair of Computer Information Systems and Computer Science at Idaho State University. He has taught both computer science and information systems courses over the course of his twenty plus years in academia. Dr. Parker's research interests include eGovernment and the elderly, business intelligence, and information assurance. He has published in such journals as *International Journal of Business Intelligence Research*, *Education and Information Technologies*, *Informing Science: the International Journal of an Emerging Transdiscipline*, *Journal of Information Systems Education*, and *Communications of the AIS*. Dr. Parker's teaching interests include web development technologies, programming languages, data structures, and database management systems. Dr. Parker holds a B.A. in Computer Science from the University of Texas at Austin (1982), an M.S. in Computer Science from Texas Tech University (1991), and a Ph.D. in Management Information Systems from Texas Tech University (1995).

Bill Davey is a Senior Lecturer in the School of Business Information Technology at RMIT University, Melbourne, Australia. His research interests include methodologies for systems analysis and systems development, information systems curriculum, eGovernment and the elderly, and information technology in educational management. Bill and Kevin have worked together co-operatively on many occasions. They have co-operated on several joint research projects and coauthored several papers relating to management information systems, programming, computers in management, eGovernment and the elderly, and IS curriculum.