Learning of Web Application Frameworks based on Concerns, Micro-Learning and Examples

Daniel Correa Botero

Universidad Nacional de Colombia

Facultad de minas, Departamento de ciencias de la computación y la decisión

Medellín, Colombia

2014

Learning of Web Application Frameworks based on Concerns, Micro-Learning and Examples

Daniel Correa Botero

Submitted in partial fulfillment of the requirements for the degree of:

Magister en Ingeniería de sistemas

Thesis Director:

Ph.D. Fernando Arango Isaza

Research Area:

Software Framework Understanding

Research Group:

Grupo de ingeniería de software

Universidad Nacional de Colombia

Facultad de minas, Departamento de ciencias de la computación y la decisión

Medellín, Colombia

2014

…To my mother, Mercedes

…To my future wife, Juliana

Acknowledgments

This master thesis would not have been possible without the support of many people. Many thanks to my thesis director, Ph.D. Fernando Arango Isaza, who spent a lot of time: making revisions, creating ideas and helping me with this project. Many thanks to Ph.D. Carlos Mario Jaramillo Zapata, who gave us excellent support: helping us to translate some papers and giving us ideas and corrections. Many thanks to Ph.D. Gloria Lucia Giraldo, who gave us support and allowed creating some experiments with her students. Thanks to Ph.D. Carlos Jaime Franco, who gave us support during the entire project to present some ideas in other countries. Thanks to Above & Beyond and BT your music is a wonderful source of inspiration. Thanks to Sebastián Gomez my friend who give me support. Thanks to Universidad Nacional de Colombia and Colciencias which provided me with the financial means to complete this project and made possible to present some chapters of this project in some conferences around the world. Many thanks to my parents they support me in all moment and their love and dedication was essential to complete this project. And finally, thanks to my sister, aunts, uncles, cousins, friends and future wife who during the entire thesis were supporting me in everything and gave me love.

Abstract

Web Applications Frameworks (WAFs) have become very popular tools for developing software applications. These tools lead to the implementation of a big amount of classes, components, and libraries which support developers for saving costs, time, and effort. Due to the big number of WAF elements, a developer needs to invest considerable effort and time in order to understand the WAF usage. Some authors had proposed different framework learning techniques, but these techniques focus on how to document or show the framework information. Then, how to drive the framework learning is a developer task. Commonly, developers follow a guide containing too much information, but in some cases developers only need to learn an incomplete WAF usage. We define in this thesis a list of WAF components, a list of web application concerns and a list of examples which create a new learning technique. This technique will indicate -based on the developers' requirements- the specific elements they should know to develop their applications. Saving time and acquiring WAF knowledge.

Keywords: software development, Micro-learning, web application frameworks, framework comprehension, example-based learning.

Resumen

Los frameworks de aplicación web -o WAFs por su sigla en inglés- se han convertido en herramientas muy populares para el desarrollo de software web. Estas herramientas poseen una gran cantidad de clases, componentes y librerías que apoyan el trabajo del desarrollador; ahorrándole costos, tiempo y esfuerzo. Debido a la gran cantidad de elementos que poseen los WAFs, los desarrolladores deben invertir mucho tiempo y esfuerzo para entender cómo utilizarlos. Algunos autores han propuesto diferentes técnicas para documentar y mostrar los elementos de los WAFs, pero como guiar el aprendizaje de un WAF sigue siendo una tarea del desarrollador. En este trabajo, definimos una lista de componentes de los WAF, una lista de intereses del desarrollo de aplicaciones web, y una lista de ejemplos; que unidas crean una nueva técnica de aprendizaje. Esta técnica indica a los desarrolladores –basados en sus necesidades-, que elementos deben aprender para desarrollar sus aplicaciones. Ahorrando tiempo y adquiriendo conocimiento en el desarrollo con WAFs.

Palabras clave: desarrollo de software, micro-aprendizaje, frameworks de aplicación web, aprendizaje de frameworks, aprendizaje basado en ejemplos.

Content

List of figures

List of tables

Glossary

Abbreviation	Term
AJAX	Acronym for Asyncronous Javascript And XML. A group of interrelated web development methods used on the client-side to create asynchronous web applications.
CRM	Acronym for Customer Relationship Management
CRUD	Acronym for Create, Read, Update, and Delete.
CSS	Acronym for Cascading Style Sheets. A language used to describe the style of document presentations in web development.
ESWS	Acronym for Empirical Studies With Students.
GUI	Acronym for Graphical User Interface.
HTML	Acronym for HyperText Markup Language.
MVC	Acronym for Model-View-Controller
ORM	Acronym for Object-relational mapping. A software-programming issue in linking object-oriented code with relational databases.
PHP	Acronym for Hypertext PreProcessor. A general-purpose server-side scripting language originally designed for web development to produce dynamic web pages.
URI	Acronym for Uniform Resource Identifier.
WAF	Web Application Framework.

Introduction

This introduction is developed with the intention to provide a short brief of each thesis chapter, for an overview of the different topics discussed and the order they are treated.

Thesis Relevance

Web Application Frameworks (WAFs) are tools used by companies, governments, universities and developers. Since WAFs are considered crucial for rapid web development [1], several frameworks are available [55][59][60][61][62][63], and the topic is subject of several researches and developments [2][3][4]. However, developers have to invest considerable effort and time in order to work with these tools [13][74]. This is due to the large quantity of components, classes, functions, libraries and elements that compose them. Many times, developers have to rely in information and material found in the internet, and sometimes that information is deprecated material or portraits wrong solutions. This situation has a negative impact over the applications quality they have to develop.

Currently, there are some methodologies proposing how to document and how to show the framework documentation. However, these methodologies don't drive developer in his/her learning process. That means developers have to select by themselves the material they want to study, but sometimes, they select material that is not related with their necessities. The reality is that developers don't need to understand everything about the WAF; they only need to understand what is related with their software requirements.

For this reason, we decided to focus our research over the WAF learning environment, aiming to identify and cover the learners' main issues to improve the WAF learning experience. We developed a technique which allows learners to save time in the developing of web applications and improve the acquired knowledge over the WAFs.

State of the art

In chapter 1, we analyze and discuss the related work over framework understanding and WAFs studies. In the first part “Framework understanding” we discuss some crucial learning statements and recommendations extracted from the literature, as follows: (i) a minimal documentation that is task-oriented helps users to faster growth in learning; (ii) examples are an effective learning strategy, especially for those beginning to learn a framework; and (iii) an important area for framework documentation is “how to use it”.

The second part “WAFs studies” shows some WAFs comparison studies, these studies established some similarities between different WAFs. Those similarities support that general WAF learning techniques can be uniformly applied to different WAFs. Besides, some WAF security studies show the importance of integrating security over the entire WAF learning, in order to create quality applications.

Research Problem

In chapter 2, we identify some specific unsolved issues and challenges in the WAF learning domain. The main issues are: (i) learning a new WAF continues being a difficult task, (ii) good documentation is difficult to find and is often outdated, (iii) WAF novice learners have to drive their own WAF learning –despite of their lack on WAF knowledge–, and (iv) WAF documentation material is limited. Based on these issues we define a thesis statement and later a thesis hypothesis, as follows: “Providing novice WAF learners with the new WAF learning technique reduces the time they need to reuse the WAF, and increases their knowledge of the WAF”. At the end of this section four research goals are defined: (i) to guide learners WAF learning by their own concerns, (ii) to provide example materials, (iii) to define a unique documentation pattern to different WAFs, and (iv) to provide a learning tool.

Solution

Chapter 3 contains the main contributions of this thesis. Having identified the research problems and the research goals, we define a series of strategies to better understand the problems, to improve the WAF learning and to provide a solution.

- **Section 3.1 - WAF components.** This section deals with the first issue: “Learning a new WAF continues being a difficult task”. We initiate describing how nowadays the WAF learning environment is. We state that WAF learning is difficult to achieve because WAFs have many components. So, classifying and understanding how these components work is the section main objective. Next, a study over six WAFs (Codeigniter [55], Yii [59], Prado [60], MVC4 [61], Ruby on Rails [62] and Cakephp [63]) was developed. In this study we identified a common list of WAF components. These components are used to develop a wide range of applications in all WAFs. Besides, we established a list of micro-tasks in order to learn how to use each component in any WAF. These micro-tasks describe in a very low level how each component is composed and what are the specific elements they use. Finally, we represented over Codeigniter a specific component with their specific micro-tasks. The main ideas of this chapter are: (i) to understand the WAF learning environment and how WAFs are composed, (ii) to classify WAF main components, and (iii) to define a list of micro-tasks which describe of components are composed and they work.

- **Section 3.2 - Web application concerns.** This section deals with another issue: “WAF novice learners have to drive their own WAF learning –despite of their lack on WAF knowledge–”. We initiate the section studying the reasons that motivate developers to learn how to use a WAF. We highlighted some important issues, as follows: (i) no matter the reason, the final goal for learning a WAF usage is to develop specific web applications, (ii) when developers have different requirements they have different learning interests or concerns, and (iii) a developer should focus in the WAF material that supports his/her interests or concerns. Based on these statements the author develops a new web application concern list and connects this list with the WAF components and micro-tasks previously described. The main idea is to define a simple way to filter the WAF material that is related with the developer concerns.

- **Section 3.3 – Introducing the use of examples.** This section deals with the last two issues: “good documentation is difficult to find and is often outdated” and “WAF documentation material is limited”. We initiate the section by introducing the importance of good examples. The use of examples have the following benefits: (i) they can reduce the amount of typing required to complete a task, (ii) finding existing

examples that match requirements, can serve as a base of code-reuse, and (iii) they can portraits the WAF architecture and help to understand how the WAF components and elements are connected. At the end we present a list examples connected with the previous concerns.

- **Section 3.4 – Defining the new WAF learning technique.** In this section we present the new WAF learning technique. In order to provide the learning path or the different learning steps, we introduce micro-learning. Micro-learning highlight the point that no matter how learning is conceptualized, in all cases there is the possibility of considering the learning process in terms of micro, meso and macro tasks. Based on this concept we state the learning steps in the new WAF learning technique: (i) in the first step the learner extract his/her application requirements and select the web application concerns related with his/her requirements, (ii) the corresponding micro-tasks documentation to each component related with each concern is presented to the learner, the learner has to read and follow this documentation in order to acquire WAF knowledge and understand the WAF components, (iii) parallel to this, for each concern a meso-task –example– documentation is also presented, the learner has to read the micro-tasks and codify the meso-tasks in order to obtain more knowledge, and (iv) the learner has to develop his/her own application –the macro-task–. At the end of this section, we present the design of a web application to support the new WAF learning technique. The main objectives of the application are: (i) provide a mechanism to complete the micro and meso tasks –these must be completed by senior WAF developers–, (ii) facilitate the access the learning material, (iii) establish a mechanism to allow learners to select their concerns and present the specific learning material to each of them.

Laboratory case

In chapter 4, we present a quasi-experiment developed with the intention to provide statistical relevance to our main research hypothesis. In this experiment three groups are defined, each group has its own treatment (see section 4.2.1). The first group baseline will use the common WAF documentation materials –cookbooks–; the other two groups will use material from the new learning technique –one group with the complete material,

another group only with the meso-tasks material–. A complete data analysis is also developed and the thesis hypothesis is confirmed.

Conclusions

In chapter 5, we present the conclusions of the research. We define a summary of the thesis main ideas, highlighting the key contributions from this proposal –explaining in detail each contribution–. At the end some future works are proposed.

1. State of the art

This chapter is motivated by analyzing and discussing the related work over framework learning. The first part “Framework understanding” portraits some crucial learning statements and recommendations extracted from the literature. These statements and recommendations are discussed for framework in general –not applied specifically to WAFs–.

The second part “WAFs studies” has been focused specifically over WAFs. Due to the lack of WAF learning studies, we collect literature about WAFs studies in general. These studies portrait important ideas and elements to develop the new WAF learning technique.

1.1 Framework Understanding

Over the past twenty years, a large range of candidate documentation techniques has been proposed to support framework understanding, including patterns [15], example-based learning [16], cookbooks [17], and visualizations [18]. Still, there is a lack of insight into problems that limit the comprehension and reuse of software frameworks. There is no true awareness of the impact these techniques have on framework understanding. As such, a few studies were conducted and their results identify some concerns and basis for future research [19]. This section will show some of these studies and some of the documentation techniques proposed to support framework understanding. The relevant ideas of each study are highlighted and are used as a base to the definition of the new WAF learning technique.

1.1.1 Frameworks

The basic processes of the software engineering are: specification, design and implementation, verification, validation and management [5]. A software developer needs tools and knowledge to develop a design and implementation of a software product. In

recent years, frameworks have become very popular tools for software development. They are powerful techniques for large-scale reuse helping developers to improve quality and save costs and time [6][7]. Nowadays they are considering crucial for rapid web development [8].

Types of Frameworks

One of the most used classifications for frameworks is Taligent classification [9]. In this classification frameworks are grouped in three categories: application, domain and support.

- Application frameworks: Application frameworks aim to provide the full range of functionality typically needed in an application. This functionality usually involves things like a GUI, documents, databases, etc.

- Domain frameworks: These frameworks can be helpful to implement programs for a certain domain. The term domain framework is used to denote frameworks for specific domains. An example of a domain is banking or alarm systems. Domain specific software usually has to be tailored for a company or developed from scratch. Frameworks can help reduce the amount of work that needs to be done to implement such applications. This allows companies to make higher quality software for their domain while reducing the time to market.

- Support frameworks: Support frameworks typically address very specific, computer related domains such as memory management or file systems. Support for these kinds of domains is necessary to simplify program development. Support frameworks are typically used in conjunction with domain and/or application frameworks.

This research focus on learning of software development applied to a full range of functionalities needed in an application. For this reason we focus on learning of application frameworks. Being precisely, we'll focus in web application frameworks (WAFs). One the most important fact is that the resultant product of a WAF is accessible from internet –web application– [10] which makes them in powerful and important tools.

1.1.2 Studies on framework understanding

The next studies provide relevant support to the thesis; each study provides important elements to be considered:

- Kirk et al. conducted three case studies to study the problems encountered by software developers when using a framework [14]. They identified general kinds of questions such as finding out what features are provided by the framework and understanding how classes communicate together in the presence of inversion of control and subtle dependencies. The authors observed that different types of documentation provided answers to a subset of the questions.

- Carroll et al. observed users reading documentation and found that the step-by-step progress induced by traditional documentation such as detailed tutorials and reference manuals was often interrupted by periods of self-initiated problem solving by users [67]. Indeed, users ignored steps and complete sections that did not seem related to real tasks, and they often made mistakes during their unsupervised exploration. Because this active way of learning was not what the designer of traditional documentation intended, Carroll et al. designed a new type of documentation, the minimal manual, that is task-oriented and that helps the users resolve errors.

- Robillard conducted a survey and qualitative interviews in a study of how Microsoft developers learn APIs [68]. The study identified obstacles to API learning ability in documentation such as the lack of code examples and the absence of task-oriented documentation. Forward and Lethbridge conducted a survey with developers and managers, and asked questions regarding the use and the characteristics or various software documents [69]. According to the participants, the following properties of software documentation were the most important: content (information in the document), upto-dateness, availability, use of examples, and organization (sections, subsections, index).

- Nykaza et al. performed a study over the desired and required content of the documentation of a framework developed by a software organization [70]. The authors observed that junior programmers with deep knowledge of the domain and senior

programmers with no knowledge of the domain had similar documentation needs about the framework. The programmers preferred simple code examples that they could copy and execute right away (as opposed to complex examples showing many features at once) and a manual that had self-contained sections so users could refer to it during their exploration (as opposed to manual that must be read from start to finish).

- Jonhson [71] identified three important areas for framework documentation to address: purpose, how to use, and design. He argued that the purpose of the framework and its constituent parts should be stated so that developers may select the correct parts for a task. While knowledge of how those parts are expected to operate allows them to be employed correctly, a description of the underlying design provides developers with an understanding of how to adapt and extend the framework in a manner consistent with the existing structure.

- Schull et al. [16] presented an evaluation of the role that examples play in framework reuse. Their study compared two approaches to framework reading and, eventually, its documentation: example-based approach and hierarchical-based approach. Their results suggested that examples are an effective learning strategy, especially for those beginning to learn a framework. They also identified potential problems with an example-based approach: finding the small pieces of required functionality in larger examples; inconsistent organization and structure of examples; and lack of design choice rationale in example documentation. They also discussed the possibility that developers become too reliant on examples and do not understand the system at a sufficient level of detail, as to implement it effectively from scratch, if necessary.

- Fayad et al [72] claimed that different alternatives could improve framework understandability: (i) refining the framework's internal design, (ii) using methods that can ensure a successful development and usage of frameworks, (iii) adhering to standards for framework development, adaptation, and integration, and (iv) producing comprehensible framework documentation. These guidelines are mainly preventive and don't focus on the issue of reusability, posing merely as general advices.

- Ho et al [73] paper presented a novel way of investigating the different philosophies for framework documentation. The philosophies included minimalist, patterns-style and extended javadoc (Jdoc) documentation. Using a survey of 90 intermediate users engaged in Command and Adaptor design patterns coding work, the exploratory study discovered that minimalist documentation has positive impacts in encouraging knowledge acquisition, significantly in terms of the framework functional workings. This concludes that documentation solutions with the minimalist principle can lead intermediate users to faster growth in learning two of the design patterns.

1.1.3 Techniques for framework understanding

During the last years, different authors have proposed different framework documentation techniques. The idea with these techniques is to produce and enhance the existing documentation with other type of information that could be used for different learners. These techniques try to represent the different framework processes and behaviors in different ways that might help to using and understanding the framework. Next, a brief summary of some proposal techniques are presented.

Cookbooks

Cookbooks are commonly used as a documentation technique for web-based framework development. Cookbooks are designed to be carefully read by programmers as reference manuals. Cookbooks also describe the entire framework composition.

- Confronting the challenge of communicating how to use the Model-View-Controller framework in Smalltalk-80, Krasner and Pope [17] built an 18-page cookbook that explained the purpose, structure, and implementation of the MVC framework. This cookbook was designed to be read from beginning to end by programmers and could also be used as a reference.

The problem with this technique is that developers have to read from beginning to end the complete material. Commonly cookbooks are plenty of pages with a big amount of information, and the reality is that developers don’t need to understand all the material. Therefore, most of them have a lack of examples; because they focus on describing in great detail how the framework is designed.

Hooks

- Froehlich et al.'s hooks [73] focus on documenting the way a framework is used, not the design of the framework. They are similar in intent to cookbook recipes but are more structured in their natural language. The elements listed are: name, requirement, type, area, uses, participants, changes, constraints, and comments.

Similar to cookbooks, hooks have a lack of good examples. The interesting point is their suggestion about focusing on documenting the way a framework is used. A learning material that shows how different elements of the framework are used and how these elements are connected these elements could be so valuable to a better framework understanding.

Patterns

- Jonhson's patterns [15] suggest documenting a framework by using a pattern language. In this language, each pattern describes a recurrent problem in the domain covered by the framework, and then describes how to solve that problem. Its main goal is to teach how to use the framework, and then complement the task-oriented information with explanations about how the framework works, for those willing to know the details. This technique tries to strike a balance between prescriptive information (how-to-do) with descriptive information (how-it-works) as to reach a larger audience of different experience levels.

- Flores [19] presents an approach to guide the framework learning process. His study presents DRIVER, a platform to teach how to use a framework in a collaborative environment. In such platform, learners can search and rate available knowledge and get recommendations for the best course of action. In this approach, learners should decide by themselves—with no guidance based on their needs—on the way they want to follow the documents.

Nowadays, WAFs present a lack of different documentation types. Commonly WAFs only support developers learning with a cookbook or web tutorial. Similar to cookbooks and hooks, patterns don't drive the developer in his/her learning; each means, developers have to figure out how to use the documentation.

Visualizations

- Jackson et al. [18] support the programmers in understanding the framework code by providing animated visualizations of example programs interacting with the framework. Commonly these visualizations show over class diagrams how the different frameworks objects are connected and represented over the framework architecture. However, a comparison with other methods is not provided.

The problem with the whole set of techniques is that even when each technique highlights valuable insights; in real WAFs documentations they are not present. Also, there is a lack of deeper studies about the validity of how each technique improves –or not– the framework learning. Also, there is a lack of comparison studies between the techniques.

1.2 WAFs

As mention before there is a lack of WAF learning studies. However, we consider relevant to collect and discuss what the recent researches are over WAFs. These researches can highlight important issues, ideas and concerns that authors have nowadays. We expected to find similarities between different WAFs that support the development of a unique technique for WAF learning, which could be applied to different WAFs no matter the programming language or the internal structure.

Web application frameworks

WAFs typically provide core functionality common to most web applications, such as user session management, data persistence, and template systems. By using an appropriate WAF, a developer can often save a significant amount of time building a web application. Most WAFs (e.g. CakePHP, Spring, Prado, and Ruby on Rails) offer websites, forums, blogs, plugins, bug fixes, and much more. But the large amount of information not necessary means a good quality of WAF material for learning.

1.2.1 WAF studies

Over the past ten years, a large quantity of research in WAFs has been done due to its importance for web development. Authors have focused their research in different areas but mainly in: WAF tutorials, WAF comparisons and WAF security aspects. Each area

portraits relevant information to the thesis. At the end there is a summary which highlights the main aspects we considered in this thesis.

Comparison studies

At first sight, each WAF seems to be independent and unique, but actually many of them have much in common. In fact, several framework comparison studies show many components and similarities between them.

- Canales [11] project examined two WAFs: CakePHP and Symfony. The project studied the structure, differences, and similarities of each framework and used that knowledge to choose a framework to begin the development of an online language placement exam template. At the end a framework was chosen, and the research continued on that framework in the form of additional reading and tutorials. Finally, a basic exam template prototype was developed.

- In Wang [12] thesis, was conducted a general comparison of four popular Java web frameworks: Struts1.X, WebWork2.2X, Tapestry 4, JSF1.2. The main idea was to try to help web developers or technique managers to gain a deep insight of these frameworks through the comparison and therefore be able to choose the right framework for their web applications. At the end an evaluation was established with the pros and cons of different WAFs features and a general suggestion of web application types that the four chosen Java web frameworks can effectively fit in.

- Plekhanova [65] report considered many factors in order to evaluate three different WAFs. Based on the factors and the experience acquired, a set of seven evaluation items was developed. These items were evaluated below on a scale of 1.00 (Poor) to 5.00 (Excellent). At the end Django received the highest weighted score of 4.05. Ruby on Rails is second with 3.85 while CakePHP got 2.95.

- Björemo and Trninić [66] created a report which looked closer at some of WAFs (CakePHP, Grails, Ruby on Rails, Stripes, Spring Roo and Wicket) to see what they had to offer and how they did it. The frameworks were evaluated based on six criteria: documentation and learning, convention over configuration, integrated development environment, internationalization (localization), and user data input validation and

testing. The conclusions were that there is no superior WAF and one should not learn a new programming language just for using a recommended web framework.

The previous comparison studies shows WAFs have some similarities; these similarities are used in a first study to better understand how WAFs are composed. This statement allows us to perform a study for WAF learning that could be applied to different WAFs no matter the programming language or the internal structure.

Security aspects

Nowadays, web applications contain many security vulnerabilities. Web applications are also widely accessible and often serve as an interface to large amounts of sensitive data stored in back-end databases. Due to these factors, web applications have attracted much attention from cyber-criminals. Attackers commonly exploit web application vulnerabilities to steal confidential information or to host malware [76]. Vulnerable WAFs applications generate a risky impact over the entire application and their users. Investigate how WAFs support –or not– web application security, what are the common vulnerabilities and how to implement this information into the new WAF learning material is the section main objective.

- Roberts-Morpeth and Ellman [75] report investigated whether a vulnerability found in one web framework may be used to find a vulnerability in a different web framework. To test this hypothesis, several open source applications were installed in a secure test environment together with security analysis tools. Each one of the applications were developed using a different software framework. The results show that a vulnerability identified in one framework can often be used to find similar vulnerabilities in other frameworks. Cross site scripting security issues are the most likely to succeed when being applied to more than one framework.

- Robertson and Vigna [76] presented a framework for developing web applications that, by construction, are invulnerable to server-side cross-site scripting and SQL injection attacks. They demonstrated that all dynamic data that is contained in a document generated by a web application must be subjected to sanitization. Similarly, we show that all SQL queries must be executed in a safe manner.

- Scholte et al [77] Shows that web applications are also frequently targeted by attacks such as XSS and SQL injection. They presented an empirical study of more than 7000 web application vulnerabilities and more than 70 web application development frameworks with the aim of gaining deeper insights into how common web vulnerabilities can be prevented. Their findings suggested that many SQL injection and XSS could easily be prevented if web languages and frameworks would be able to automatically enforce common data types such as integer, Boolean, and specific types of strings such as e-mails and URLs.

- Jayaraman et al [79] described a new approach for enforcing request integrity –such as: Cross-site-request forgeries (CSRF) and workflow violations– in a web application and its implementation in a tool called Bayawak. Under their approach, the intended request sequences of an application are specified as a security policy. And a framework-level method enforces the security policy strictly and transparently without requiring changes in the applications source code.

The recent research shows some concern in WAF security aspects. Many tools and applications to prevent different attacks have been developed. But, the reality is even when most WAFs have some components to prevent the common vulnerabilities like: CSS attacks, SQL-injections, and CSRF; some applications continue being vulnerable. The problem could be in the way some developers use WAFs components. This aspect could be due to the developers' lack of knowledge on security aspects. Also, could be due to the WAFs documentation, which presents big amount of information in which could be difficult to find the proper components to prevent these attacks.

How to develop documentation easy to read that at the same time contains security aspects, is one challenge of this thesis.

1.3 Summary

In the first part, some important studies on framework understanding have been done. Besides, some techniques for framework understanding like: patterns, example-based learning, cookbooks, and visualizations were analyzed. These studies highlight some important aspects to this thesis: (i) a minimal documentation that is task-oriented helps users to faster growth in learning; (ii) examples are an effective learning strategy,

especially for those beginning to learn a framework; and (iii) an important area for framework documentation is “how to use it”. It’s important to highlight than even when there are many framework documentation techniques proposed, most of these techniques are not used in real WAFs documentations.

The second part shows most WAFs studies have been focused on: WAFs tutorials, WAFs comparison and WAFs security aspects. The WAFs comparison studies shows some similarities between different WAFs, these similarities support a study in WAF learning in general that could be applied to different WAFs no matter the programming language or the internal structure. The WAFs security studies show the importance that security has nowadays. These studies shows that even most WAFs have components and elements to prevent different attacks, some developers don’t use them in their applications. Improve security integration over the WAF learning appears as an important concern.

2. Research Problem

Learning of WAFs deals with a lot of aspects, but mainly, with understanding the WAF elements and how to use them to develop different applications. In this chapter, several open research issues are raised focusing on WAF understanding and opportunities for improving the existing WAF learning techniques; and creating a new WAF learning techniques are identified.

2.1 Open issues

From the state-of-the-art review presented in the previous chapters, a number of open research issues arise. An insight of the most relevant ones follows, in order to focus the scope of the work presented in this thesis:

- **Learning a new WAF is difficult to achieve:** learners have to invest considerable effort and time in order to work with frameworks in general. This is due to the large quantity of components, classes, functions, libraries and elements that compose them. Frequently WAFs are considered very complex: (i) very abstract; (ii) plenty of documentation, hundreds of pages that maybe you're not going to use; (iii) obscure, in the sense that it usually hides existing dependencies and interactions between classes [19]. In the case of novice developers, they lack the needed experience and ignore what WAF facilities are available to them, so they do not know what to look for. Another issue is each WAF defines its own documentation strategy, making difficult for a new WAF leaner to find the proper documentation over different WAFs. These problems makes learners spend considerable amount of effort to understand and learn how to use a WAF.

- **Good documentation is difficult to find and is often outdated:** nowadays if a learner wants to work with a specific WAF, he/she has two options: look for documentation in the WAF official website, or look for documentation in other

websites. The first site usually provides a cookbook to work with, but maybe learners want to access examples or to answer specific questions. The problem with other websites is that they could show deprecated information, wrong examples and could affect the learner software quality.

- **How to drive the WAF learning is a learner task:** the actual WAF documentation methodologies don't drive developer/learner in his/her learning. It means developers have to select by themselves the material they want to learn, but sometimes, they select material that is not related with their necessities. The reality is that developers don't need to understand all WAF usage; they only need to understand what is related with their software requirements.

- **WAF documentation material is limited:** Currently, there are some methodologies which proposed how to document and show a framework document [15][16][17][18]. But the reality is WAF creators only create cookbooks, despite of the other methodologies that could be useful for different learners. Also, techniques for WAF understanding are still not studied in detail.

2.2 Research questions

From the aforementioned open research issues, a few research questions revolve around a major question that is considered central to the presented research work: How to improve WAF learning? Those questions are listed next.

- Do WAFs share characteristics between them? What are the similarities between different WAFs? Is it possible to establish a list of WAF common components? (see section 3.1)
- What are the WAF learner goals? Where do they start? What do they look for? What are the learners concerns? (see section 3.2)
- What kind of documentation materials could serve as a base to improve the WAF learning? (see section 3.3)
- How to connect the learners concerns with the specific WAF material? How to drive learners in their WAF learning? (see section 3.4)

2.3 Thesis statement

Based on the research challenges presented before (sections 2.1 and 2.2) and the state-of-the-art review (Chapter 1), we state that:

"Providing a new WAF learning technique that focus on the specific learner concerns and provide a specific learning material composed by micro documentation and examples will allow WAF novice learners, to acquire WAF knowledge and develop their application in less time than with the common learning materials"

What is meant by "a new learning technique"? A new method composed by different steps and materials which helps a learner to drive their own learning.

What is meant by "micro-documentation"? A documentation of an atomic element of a specific WAF (see section 3.1.3).

How the knowledge acquired by learners is measured? In chapter 4 a quasi-experiment is developed. The knowledge acquired is measured when learners are submitted to a post-questionnaire which contains some WAF questions.

Who are the novice WAF learners? Any developer who never had developed applications by using a WAF.

How time is measured? In chapter 4 a quasi-experiment is developed. Time is measured by the completion of different tasks. In which some learners are divided in groups, some of them had to use the common learning materials and other the new learning technique materials. At the end time is compared.

What are the "common learning materials"? Usually for WAF learning, WAFs only provide a cookbook or a web tutorial plenty of documents that indicates how the WAF works and how to use it.

2.4 Research hypothesis

The previous thesis statement could be redefined as the follow hypothesis:

H: Providing novice WAF learners with the new WAF learning technique reduces the time they need to reuse the WAF, and increases their knowledge of the WAF.

2.5 Research goals

This thesis aims at contributing to the body of knowledge in software engineering. Concretely, it strives to improve WAF learning for novice developers, by driving the developer WAF learning and giving a specific learning material to him/her. This will be achieved in four ways:

1. By guiding learners WAF learning by their own concerns: Before to start using the learning material, a learner has to identify their own concerns. Then, he/she has selected his/her related concerns and the specific related material is given to him/her.

2. By providing example materials: Each developer concern will have an associated code-example that will serve as a base to develop his/her own applications and that will serve as a base of code reuse and as a base to understand how the WAF components are connected.

3. By defining a unique documentation pattern to different WAFs: by using the same documentation strategy, developers can find a specific learning material over different WAFs without spending too much effort.

4. By providing a learning tool: A web application tool will be developed in order to facilitate the documentation completion and serve as a tool to drive the developer WAF learning.

2.6 Summary

WAF learning is a complex task; we identified some specific issues and challenges in this domain: (i) WAF learning continue being a difficult task, (ii) good documentation is difficult to find and is often outdated, (iii) WAF novice learners has to conduct their own WAF learning –despite of their lack on WAF knowledge–, and (iv) WAF documentation material is limited.

Based on these issues four main goals were identified: (i) guiding learners WAF learning by their own concerns, (ii) providing example materials, (iii) defining a unique documentation pattern to different WAFs, and (iv) providing a learning tool. Later in chapter 4 the author proposes to validate the proposed goals through developing of a controlled (quasi-)experiment, performed in academic contexts.

3. Solution

This chapter contains the key contributions of this thesis. It defines the principal elements that are used in the new WAF learning technique. Each section of this chapter tries to resolve a research problem (see section 2.1). At the end all these elements are combined into the new WAF learning technique (see section 3.4), as a whole it defines the path novice learners should follow in order to learn to use a new WAF.

3.1 WAFs Components

3.1.1 Introduction

This section deals with the first issue: “Learning a new WAF continues being a difficult task”. We initiate describing how nowadays the WAF learning environment is. We state that WAF learning is difficult to achieve because WAFs have many components. So, classifying and understanding how these components work is the section main objective. Next, a study over six WAFs (Codeigniter [55], Yii [59], Prado [60], MVC4 [61], Ruby on Rails [62] and Cakephp [63]) was developed. In this study we identify a common list of WAF components. These components are used to develop a wide range of applications in all WAFs. Besides, we establish a list of micro-tasks for learning how to use each component. These micro-tasks described in a very low level how each component is composed and what are the specific elements they use. Finally, we represent over Codeigniter a specific component with their specific micro-tasks. The main ideas of this chapter are: (i) to understand the WAF learning environment and how WAFs are composed, (ii) to classify WAF main components, and (iii) to define a list of micro-tasks which describe of components are composed and they work.

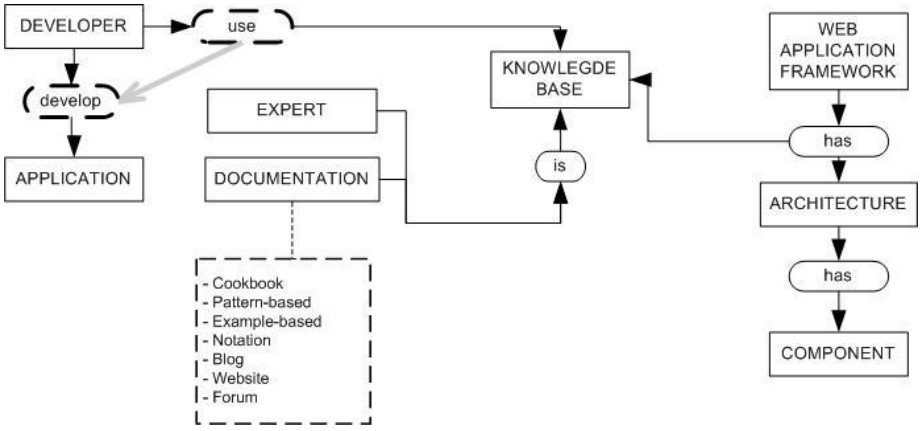
Establishing a list of WAFs main components and their micro-tasks, serves as a base to define a unique documentation pattern to different WAFs.

3.1.2 WAF learning environment

Deep WAF knowledge is difficult to achieve because these tools have many components and elements. Most WAFs have for example a role manager component –which allows managing the application permissions–; an error handler component –which allows capturing and displaying properly the information errors–, a route manager –which establish the communication between the different framework layers–, and a cache component, among others. These components are crucial to software reuse techniques, but too many components are involved in complex WAF development technologies [6][7][38].

In Figure 3.1 we use the so-called pre-conceptual schemas [39] for representing the actual WAF learning environment. Sometimes, the only guidelines for developers are the official documentation, regarding other knowledge bases from which they can extract information. In other cases, developers are assigned to an expert developer or a partner who guides him/her in the learning process. In such cases, time-usage and teaching-based constraints leave the novice developers unguided.

Figure 3-1: A representation of the WAF learning environment.



By pointing the key components the developer have to look for—during WAF understanding process—and by presenting the learning tasks associated with each element, we expect to significantly improve the WAF learning process.

3.1.3 Establishing WAF Components and Micro-tasks

During the 2013, we built seven applications —e.g., currency converter, create-read-update-delete (CRUD) facilities for several database management systems, a light course

application, data validations, and so on— in various WAFs (Codeigniter [55], Yii [59], Prado [60], MVC4 [61], Ruby on Rails [62] and Cakephp [63]) in which we covered a diverse set of concerns (described at section 3.2). No matter we use different WAF components in order to build similar applications we recognize we are using the same purpose-oriented components, such as: error handler, data validation, role manager, ORM, AJAX, auto-code generators, and template manager, among others. Consequently, the WAF facilities should be considered the same. Their differences were essentially related to syntax and WAF functionalities.

After these studies, we decided to create a WAF generic components list. First, we decided to use a unique name for similar components. For example, the component responsible for establishing a device for accessing the methods or functions of a controller (routes) in the Yii framework is called "URL Management," while in Codeigniter is called "URI Routing." Instead of identifying those components by their proper names, we decided to call that component "Route Manager". By looking to our list a developer working with an unknown WAF can understand what components are shared by other WAFs and what components are new.

Figure 3.2 shows an example of the done process. This was a piece of code of a login system developed in Codeigniter WAF, in this piece of code were identified 4 key basic micro-learning-tasks (for simplicity micro-tasks) that a developer should read and follow in order learn how to develop his/her application, these micro-tasks are related with a specific WAF generic components.

Figure 3-2: An example of Codeigniter components and micro-tasks identification.

Based on the above statements we established 13 WAF main components [2]. Each component has different micro-tasks (see example figure 3.3). These micro-tasks were created based on our experience developing software applications in WAFs at the university, and using WAFs for real projects; they defined at very low level the crucial elements of each component and how they work. Table 3.1 includes WAF main components, its description, and the associated micro-tasks.

Figure 3-3: A component with its specific micro-tasks.

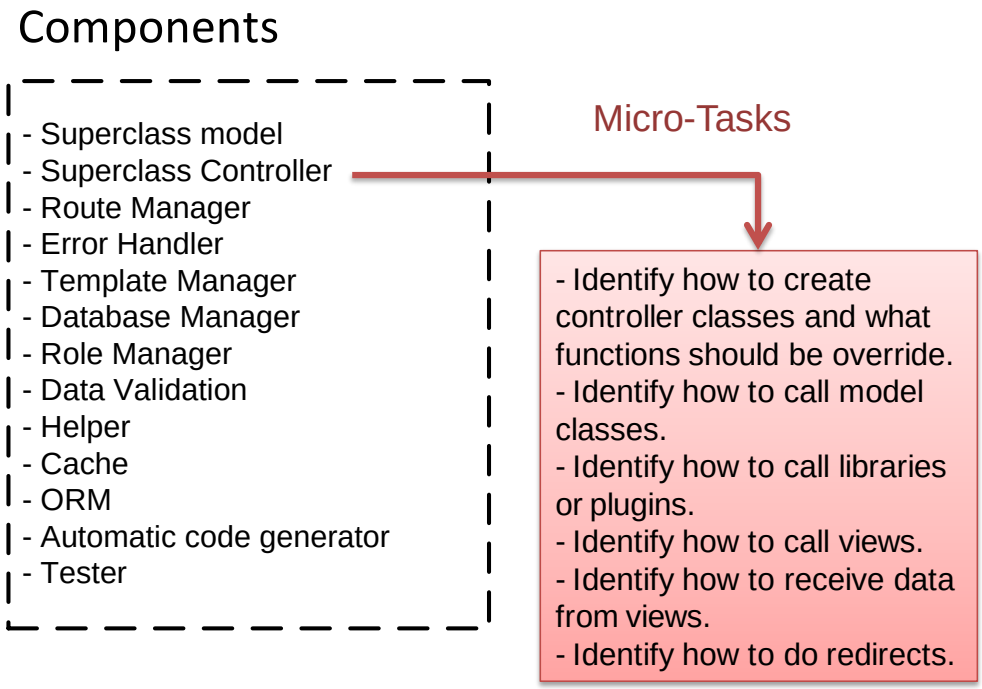


Table 3-1: WAFs Components.

Component	Description	Micro-Tasks
Superclass model	It provides a list of useful methods, functions and variables can be used by models for extension purposes.	- Identify what functions are available - Identify how to create model classes and what functions should be override - Identify how to create new class functions - Identify how to call attributes and functions classes

Component	Description	Micro-Tasks
Superclass Controller	It provides a list of useful methods, functions and variables can be used by controllers for extension purposes.	- Identify what functions are available - Identify how to create controller classes and what functions should be override - Identify how to call model classes - Identify how to call libraries or plugins - Identify how to call views - Identify how to do redirects - Identify how the variables get, post, session, and files are treated - Identify how to receive and send data to views - Identify how to show results by pages - Identify how to manage different packages of languages - Identify how to show information depending on user's location - Identify how to manage login and logout - Identify how to upload files - Identify how to design an application for desktop and mobile
Route Manager	It establishes a device for accessing controller methods or functions	- Identify how URLs are and what means each part of the URLs - Identify how to send and receive data from URLs
Error Handler	It defines the way to catch and show the errors.	- Identify what the sections to catch errors are - Identify what the types of errors are - Identify how to capture and show these errors
Database Class	It defines the way for accessing, editing, or saving information into the database by using controllers and objects.	- Identify how to connect to a specific database - Identify how to add data to the database - Identify how to delete data from the database - Identify how to edit data from the database - Identify how to filter data - Identify how to select data from the database (even information from various tables) - Identify additional functions or functionalities
Template Manager	Also called "template engine," it provides communication bridges between controllers and views and defines some functions and special syntax in both layers.	- Identify if a different syntax is used in the view layer and how it works - Identify how the communication between controller and view layers is achieved - Identify what functions are available - Identify how the variables get, post, session, and files are treated - Identify how to create styles (css files) and where are located
Role Manager	It provides a way to verify whether or not a user is granted to manipulate specific resources, or whether he/she is allowed to enter to specific zones.	- Identify how to validate permissions in the application - Identify how to grant access to specific areas. - Identify how to add types of roles

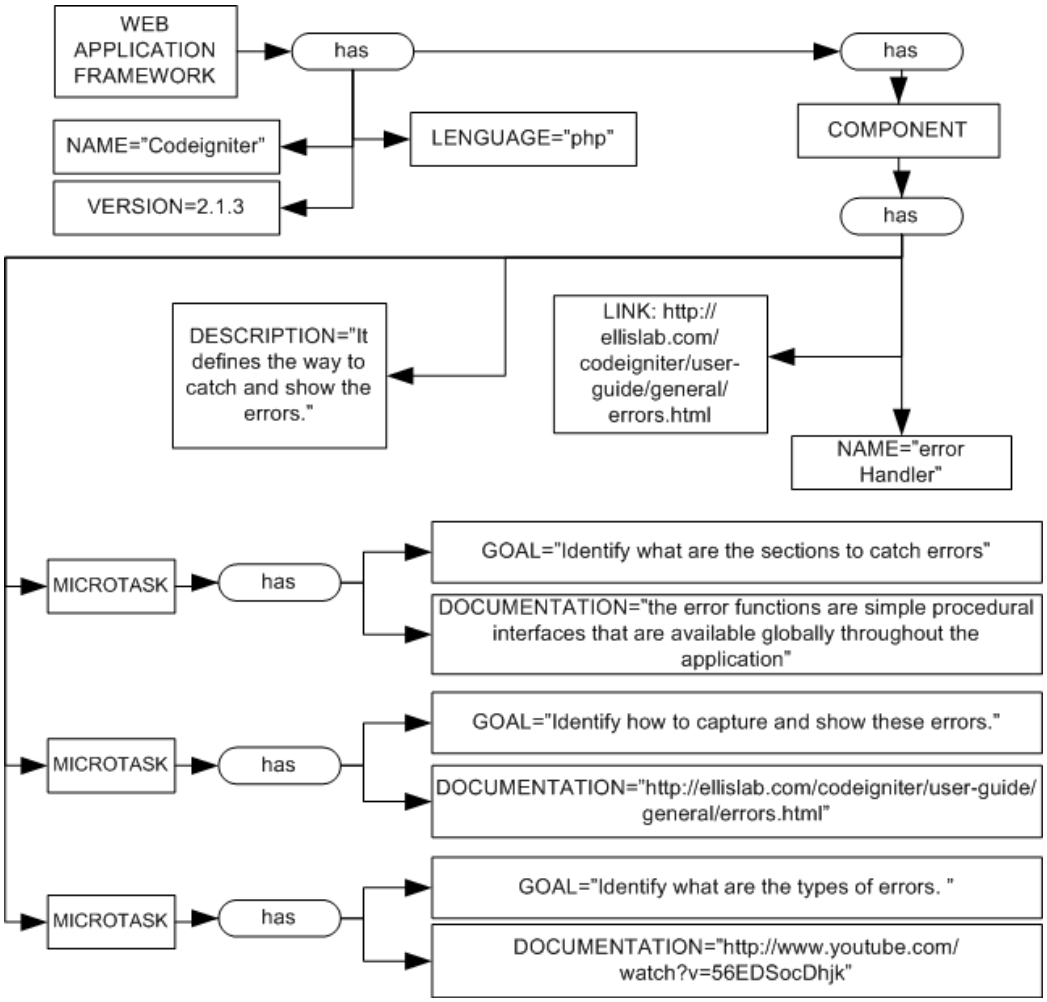
Component	Description	Micro-Tasks
Data Validation	It defines how to validate that information in objects or variables is right. In some cases this component is associated with the model layer. Besides, sometimes it defines a list of functions or elements to check the right type of variables.	- Identify how validations in control layer are treated - Identify how validations in view layer are treated - Identify how validations in model layer are treated - Identify what kinds of validations are predefined - Identify how to create new validation types
Cache	It defines a way of caching webpages, in order to achieve maximum performance and improve the server load.	- Identify how to call cache - Identify where cache is used
Helper	Helpers are collection of functions in a particular category. They are helpful for doing tasks. For example, some URL Helpers support the link creation and the element form creation, among others.	- Identify what kinds of helpers exist - Identify what facilities give each helper and how to use them - Identify how to create and connect a new helper or library
Tester	It provides a device to test and debug your applications, to find possible bugs, with real data or sample data. It allows you to show debugged information about the contents of variables.	- Identify how to create unit tests - Identify how to debug information
ORM	It defines a mapping between objects and relational databases. Some WAFs use their own classes (or functions in the model layer) and others use ORM programs.	- Identify how the transformation among relational databases and class objects is achieved - Identify how various objects are gathered from different classes - Identify how one-one and many-many relations, among others, are treated - Identify how to call specific SQL statements
Automatic code generator	It provides a way to automatically generate code, e.g. in some cases WAFs provide a CRUD module (create-read-update-delete). This module usually works adding information from a form.	- Identify how to call and use auto-code generators. - Identify what information is created and how to edit it - Identify how to delete that information

Representing one component on Codeigniter

The main WAF components depicted in the Table 3.1 can be exemplified by representing an actual WAF. We selected Codeigniter for this purpose.

We extend the use of pre-conceptual schemas by using the so-called executable pre-conceptual schemas [40] in order to represent our example.

Figure 3-4: A component on Codeigniter with its specific micro-tasks.



In Figure 3.4 we propose the representation of the Codeigniter "Error handler" component. In this Figure we represent the component micro-tasks and documentation. Besides, we provide some information to the developer about what he/she will find and what he/she will need to use from that component. Such information could be used as a starting point in order to acquire knowledge about how the component works and what

are its principal elements. Also, the representation provides a link to where to go for more information.

An expert can pick up some information for completing the pre-conceptual schema, and can provide such information to novice developers. Also, the use of executable pre-conceptual schemas could be useful in order to create a functional application in which the Meta model information is stored.

3.1.4 WAFs components summary

WAF learning is a complex task to achieve because these tools have many components. These components seem to be very similar no matter the WAF a developer use. We developed a research in which we built seven mini-applications covering a diverse set of concerns (described at section 3.2) in six WAFs (Codeigniter, Yii, Prado, MVC4, Ruby on Rails and Cakephp). After this process we establish that no matter we use different WAF components in order to build similar applications we recognize we are using the same purpose-oriented components. Consequently, the WAF facilities should be considered the same. Their differences were essentially related to syntax, availability and WAF functionalities. Finally, we establish a list of micro-tasks for learning how to use each component. These micro-tasks described in a very low level how each component is composed and what are the specific elements they use.

3.2 Web Application Concerns

3.2.1 Introduction

This section deals with another issue: “WAF novice learners have to drive their own WAF learning –despite of their lack on WAF knowledge–”. We initiate the section studying the reasons that motivate developers to learn how to use a WAF. Then, some important issues are highlighted: (i) no matter the reason, the final goal for learning a WAF usage is to develop specific web applications, (ii) when developers have different requirements they have different learning interests or concerns, and (iii) a developer should focus in the WAF material that supports his/her interests or concerns. Based on these statements the author develops a new web application concern list and connects this list with the WAF components and micro-tasks previously described (see section 3.1). The main idea is to define a simple way to filter the WAF material that is related with the developer concerns.

3.2.2 Developers Concerns

Developers learn to use WAFs for different reason: developing a software project, acquiring more knowledge, applying for a job position, accessing the training about tools in organizations, etc. However, no matter the reason, the final goal for learning a WAF usage is to develop specific web applications.

These specific web applications could be very different from one to another. For example:

- Developer A could be requested to develop a complex Customer Relationship Management (CRM) system.
- Developer B could be requested to develop a simple static website.
- Developer C has to develop a simple under-construction home page.

The first application –CRM system– involves a lot of requirements, more than the other applications. That means developer A has to acquire more WAF knowledge –reading and accessing more WAF information– than the other developers. We could also recognize that application B probably involves less data persistence and less database effort, and finally probably application C only involves displaying information on screen (i.e., developer C is focused on a very specific concern). In other words, different developers are driven by different interests or concerns.

This concept highlight a very important point: when developers have different requirements they have different learning interests or concerns.

Consequently, if in the learning process, a developer should focus only in the WAF material related with his/her interests or concerns, he could save time and effort in order to learn the WAF and developing the web application.

How to identify web developers' main concerns is the main objective on this chapter. Besides, how to connect these concerns with the specific WAF documentation is described later in this section.

The use of Concerns

In the software development context, a concern is a particular goal, concept, or area of interest. For example, the core requirements of a library borrow card processing system is related to processing book transactions; while its system level concerns would be handle logging, transaction integrity, authentication, security, performance, etc. [21].

This idea, of separation of concerns was since the beginning a characteristic of almost all Web methodologies, like HDM [22], OOHDM [23], etc. At the beginning, this separation of concerns was only applied to the design and implementation phases of the development process. But, nowadays we can observe a clear tendency towards a separation of concerns from the very beginning, i.e. during the requirements elicitation phase. It is interesting to remark, that the use of different terminology for the same or similar concepts made a comparison study difficult. We stress the need to standardize the terminology used in Web methodologies [24].

Some authors use concerns to create metamodels of web applications [25][26]. Kong et al [27] use separation of concerns to define perspectives of the different participants in the web application development process. Like: business owners, web system users, information architects, system architects, developers, and testers. Sousa et al [28] use concerns in Aspect-Oriented Software Development (AOSD). They use them at various levels of abstraction, from requirements (even to declare non-functional requirements like: security and performance) to design artifacts. Brito et al [29] use them to refer to a matter of interest which addresses a certain problem that is of importance to one or more stakeholders, defining a concern as a property that the future system must provide.

Based on this perspective, we could face the WAF learning by using separation of concerns. Separation of Concerns (SoC) has been used in multiples software areas during the last years, e.g., requirements specifications [30], framework architectures [27], and aspect-oriented programming [31]. SoC is a basic principle of software engineering. Derived from common sense, SoC essentially means that dealing successfully with complex problems is only possible by dividing the complexity into sub-problems which can be handled and solved separately from each other [32].

We use these separation of concerns connected to WAF components and micro-tasks, giving a specific structure of the elements that a developer should learn for supporting the application requirements.

3.2.3 Creating a new web application concern list

Some authors have defined different concern lists or methods to define concerns [27][28][29][30], but in most cases the definition of these concerns is delegated to an analyst. In other cases, the concern list is just a list of non-functional requirements or a list of high level objectives like: immunity, integrity, precision, robustness, among others.

However, these concern lists are very general and are difficult to adapt to the specific WAF components and elements that a developer should learn. So, based on the idea of driving WAF learning through a concern list, we developed a new web application concern list.

In order to develop this list, we analyzed more than 20 web projects that were develop by computer science students in a course during 2012 and 2013. These projects are based on real industry needs. We found similarities among each project requirements and we grouped them in a concern list. In this analysis we registered how many projects required a specific concern. Also, this analysis shows that no matter how different seems each application from one another, they use similar concerns.

After this process, we define in Table 3.2, 29 concerns and we categorize them in different groups [42]. At the beginning a developer has to recognize the specific requirements for the project he/she is working on. After that, he/she has to carefully read

each concern and its specific description. Finally, he has to select the concerns which are involved in his/her project requirements.

At the end of the section each concern will be connected to the specific components or elements of a WAF. This generates a personalized learning guide.

Table 3-2: Web application concerns list.

#	**Concern** **(Times of appearance on projects)**	***Category***	***We suggest to select this concern if:***
1	Display information on screen (20)	User Interface	You have to display information on a screen.
2	Stylized screens (20)	User Interface	Your screens have to be edited and stylized usually through a CSS file. Sometimes WAFs are based on prefabricated styles.
3	Tools and accessories for creating views (20)	User Interface	You have to create forms, tables, or other view elements. (Some WAF support to create faster view elements usually using front-end languages like html).
4	Routes and navegability (20)	User Interface	You need to display a screen. Each application section or link has a specific route. These routes and their connections are very different from WAF to WAF.
5	Capture and assign data (20)	User Interface	Your application involves creating forms, to capture data, or to send data from a controller to a view.
6	Client-side data validation (20)	User Interface	You need to do validation in client side like guarantee not empty forms or specific type of data or validations using AJAX. Besides, don't forget to revalidate in server-side.
7	Upload files (13)	Architecture and data flow control	You need to upload files like images, and documents, among others.
8	Error handling (20)	Architecture and data flow control	Your application generates client errors, or database errors, or any kind of errors. It is important to know how to treat them, how to capture them and show them.
9	Internationaliz ation (3)	Architecture and data flow control	Your application requires multiple languages or to have the screens texts centralized (which improves maintainability).
10	Localization (2)	Architecture and data flow	The information displayed on your application screens depends on user location (e.g., show a specific app to a

#	**Concern** **(Times of appearance on projects)**	***Category***	***We suggest to select this concern if:***
		control	user on US and another to a user in UK).
11	Caching (3)	Architecture and data flow control	Performance is a very important requirement. Some WAF use caching systems to have pre-storage of the information.
12	Testing (7)	Architecture and data flow control	You need to know how to debug the application information or to apply some test.
13	Portability (7)	Architecture and data flow control	You need to develop a version of your application for desktops and another for mobiles.
14	Data Selection (20)	Data modeling and persistence	You need to extract data from a class model (usually connected to a table of your database).
15	Data Selection with pagination (19)	Data modeling and persistence	You need to extract data by pages from a class model (usually connected to a table of your database).
16	Data selection using filters (20)	Data modeling and persistence	You need to select filtered data (usually using specific searches).
17	Multiple data selection (20)	Data modeling and persistence	You need to extract data from multiple class model (usually connected to various table of your database).
18	Data storage (20)	Data modeling and persistence	You need to save data from a class model (usually save data on your database).
19	Data editing (19)	Data modeling and persistence	You need to edit data from a class model (usually update data your database).
20	Deleting Data (14)	Data modeling and persistence	You need to delete data a class model (usually delete data your database).
21	Creating model functions (20)	Data modeling and persistence	You need to create specific functions for your classes.
22	Model-side data validation (20)	Data modeling and persistence	You need to apply model-side validations.
23	Authentication (20)	Security	You need a login in your application.
24	Authorization (20)	Security	You need to grant access to different areas in your application.
25	Control data in session (20)	Security	You need a login, a shopping cart or other functionality that require control data in session.
26	Controller-side data validation	Security	Your application require validate data (usually additional

#	**Concern** (Times of appearance on projects)	***Category***	***We suggest to select this concern if:***
	(20)		data that data from models).
27	Coupling modules (14)	Modules and extensions	You need to couple a specific module in your application (some WAFs have websites plenty of specific modules like calendars, pdf generation, transformation to csv and much more). You have to search if the module you need is available or you have to develop it.
28	Creating modules (14)	Modules and extensions	You need to create a new module in your application.
29	Auto-generated code (14)	Modules and extensions	Your WAF offers the possibility to auto-generate a CRUD (create-read-update-delete) of a class model.

Table 3-3: Web application projects vs concerns.

Project/Concern	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
Totto	x	x	x	x	x	x	x	x				x		x	x	x	x	x	x	x	x	x	x	x	x			x
MaderApp	x	x	x	x	x	x		x				x		x	x	x	x	x			x	x	x	x	x			
Gestor de fondo emp	x	x	x	x	x	x		x				x		x	x	x	x	x	x	x	x	x	x	x	x			x
HMRO	x	x	x	x	x	x	x	x				x		x	x	x	x	x	x	x	x	x	x	x	x			x
Terebotero	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Supergas	x	x	x	x	x	x	x	x				x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Agenda	x	x	x	x	x	x		x					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Calculadora Credit	x	x	x	x	x	x		x					x	x		x	x	x	x		x	x	x	x	x	x	x		
Vmaxcoffee	x	x	x	x	x	x	x	x		x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
TierraCafetera	x	x	x	x	x	x	x	x			x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Inteinsa	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
SGI	x	x	x	x	x	x	x	x				x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Credistore	x	x	x	x	x	x		x						x	x	x	x	x	x		x	x	x	x	x	x			
Licores	x	x	x	x	x	x	x	x						x	x	x	x	x	x		x	x	x	x	x	x	x	x	
Hoteles	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x	x	x			x
Joyeria	x	x	x	x	x	x		x			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Juegos	x	x	x	x	x	x	x	x		x				x	x	x	x	x	x		x	x	x	x	x	x	x		
Distrieggs	x	x	x	x	x	x		x					x	x	x	x	x	x	x		x	x	x	x	x	x	x		
Empleos en la red	x	x	x	x	x	x	x	x						x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Frameworkg	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Total	20	20	20	20	20	20	13	20	3	2	3	7	7	20	19	20	20	20	19	14	20	20	20	20	20	20	14	14	14

Table 3.3 shows the different projects and how was collected the information. A cross in the table indicates the concern was present in the project.

3.2.4 Connecting concerns with components and micro-tasks

As we see in sections 3.2.1, 3.2.2 and 3.2.3 developers should focus only in the WAF material related with their concerns. A method to define and filter the WAF material is connecting the web application concerns to the specific WAF components and their tasks.

This connection gives the possibility to know for each concern what are the specific components and micro-tasks related to start the personalized learning process, allowing filter information that is not relevant for some developers.

Table 3.4 exhibits the common connection between the lists. The connection is not an ultimate one; a senior WAF developer could make adjustments as he/she considers. Later, the main idea is a senior WAF developer define the proper documentation to each micro-task for a specific WAF (this documentation could be a link to website, forum or blog; could be a video or a specific explanation text). Later, a real example is developed.

We need to emphasize that one concern could be related to a specific task or multiple tasks, of one or multiple components.

Table 3-4: Concern List vs WAFs Components list.

Component	***Micro-Task***	***# of related Concerns***
Superclass model	Identify what functions are available	14, 15, 16, 17, 18, 19, 20, 21
	Identify how to create model classes and what functions should be override	14, 15, 16, 17, 18, 19, 20, 21
	Identify how to create new class functions	21
	Identify how to call attributes and functions classes	14, 15, 16, 17, 18, 19, 20, 21
Superclass Controller	Identify what functions are available	1
	Identify how to create controller classes and what functions should be override	1
	Identify how to call model classes	14, 15, 16, 17, 18, 19, 20, 21
	Identify how to call libraries or plugins	27, 28
	Identify how to call views	1
	Identify how to do redirects	8, 23, 24
	Identify how the variables get, post, session, and files are treated	5, 23, 25
	Identify how to receive and send data to views	5, 7
	Identify how to show results by pages	15
	Identify how to manage different packages of languages	9
	Identify how to show information depending on user's location	10
	Identify how to manage login and logout	23
	Identify how to upload files	7

Component	*Micro-Task*	*# of related Concerns*
	Identify how to design an application for desktop and mobile	13
Route Manager	Identify how URLs are and what means each part of the URLs	4
	Identify how to send and receive data from URLs	4
Error Handler	Identify what the sections to catch errors are	8
	Identify what the types of errors are	8
	Identify how to capture and show these errors	8
Database Class	Identify how to connect to a specific database	14, 15, 16, 17, 18, 19, 20
	Identify how to add data to the database	18
	Identify how to delete data from the database	20
	Identify how to edit data from the database	19
	Identify how to filter data	16
	Identify how to select data from the database (even information from various tables)	14, 15, 16, 17
	Identify additional functions or functionalities	--
Template Manager	Identify if a different syntax is used in the view layer and how it works	1
	Identify how the communication between controller and view layers is achieved	1, 5, 7
	Identify what functions are available	1
	Identify how the variables get, post, session, and files are treated	5
	Identify how to create styles (css files) and where are located	2
Role Manager	Identify how to validate permissions in the application	24
	Identify how to grant access to specific areas.	24
	Identify how to add types of roles	24
Data Validation	Identify how validations in control layer are treated	26
	Identify how validations in view layer are treated	6
	Identify how validations in model layer are treated	22
	Identify what kinds of validations are predefined	6
	Identify how to create new validation types	--
Cache	Identify how to call cache	11
	Identify where cache is used	11

Component	*Micro-Task*	*# of related Concerns*
Helper	Identify what kinds of helpers exist	3, 27
	Identify what facilities give each helper and how to use them	3, 27
	Identify how to create and connect a new helper or library	28
Tester	Identify how to create unit tests	12
	Identify how to debug information	12
ORM	Identify how the transformation among relational databases and class objects is achieved	14, 15, 16, 17, 18, 19, 20
	Identify how various objects are gathered from different classes	17
	Identify how one-one and many-many relations, among others, are treated	17
	Identify how to call specific SQL statements	--
Automatic code generator	Identify how to call and use auto-code generators.	29
	Identify what information is created and how to edit it	29
	Identify how to delete that information	29

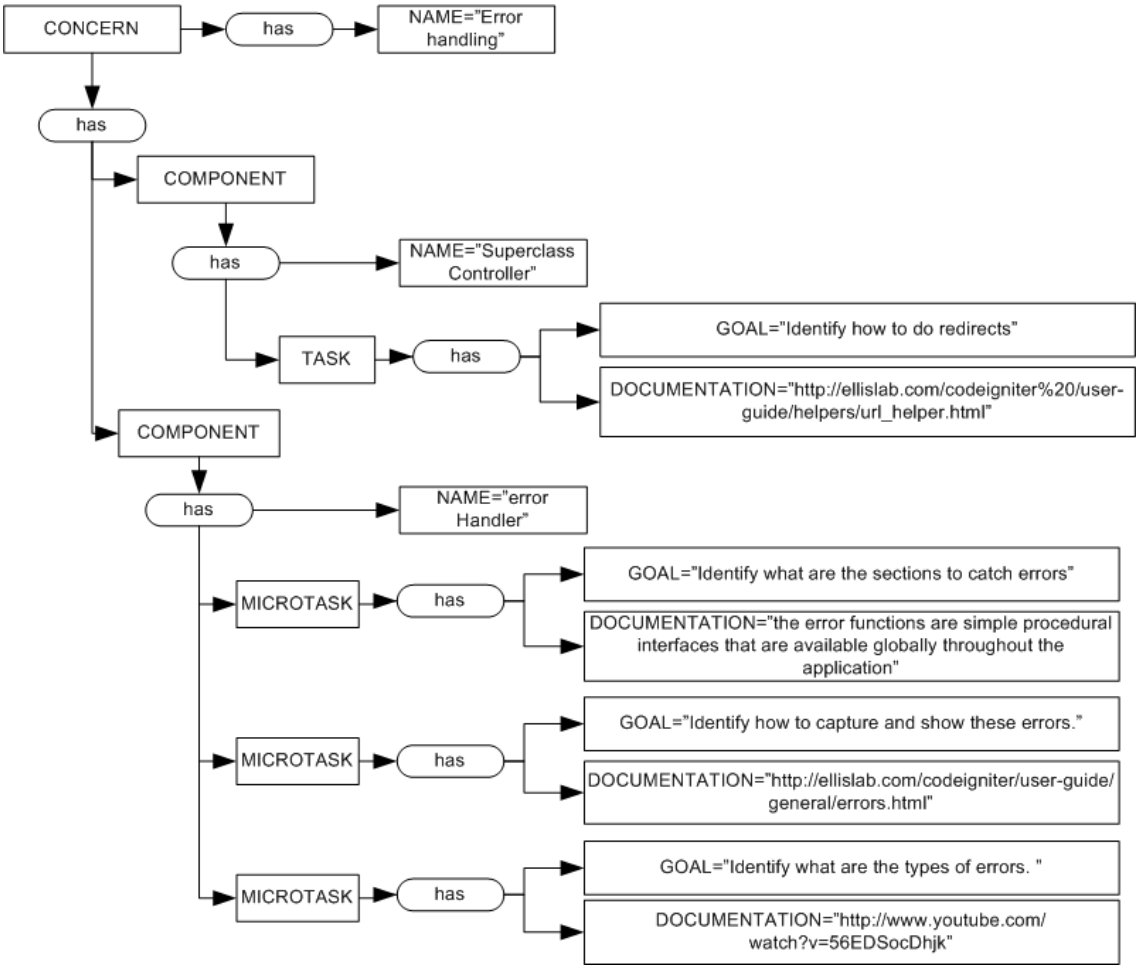
These lists also give a perspective of the components all developers should take advantage of. If a WAFs first-time user read the concern list, he/she could find crucial concerns unknown to him/her (e.g., internationalization, caching, and portability, among others). This means that if he/she implements these concerns at the beginning of the development; the final application would have more quality.

The final step, given the learning tasks, is to associate the specific learning material for each micro-task in a specific WAF. As these associations are very different for each WAF, and are out of our scope, we suggest this process should be done by a senior WAF developer –we design a laboratory case with real material on Codeigniter at chapter 4- . Additionally, we developed a web application capable to register these associations (see section 3.4.3).

Figure 3.5 is developed by using an executable pre-conceptual schema [40]. In this figure, we show an example about how concerns, components and micro-tasks are connected. If a developer is only interested on capturing and fixing errors, he/she has to read and learn micro-tasks documentation. If a developer is interested on the error handling concern,

he/she could be also interested on others concerns like: “display information on screen” or maybe “Client-side data validation”, which increase the number of components and micro-tasks he/she has to analyze and learn.

Figure 3-5: A concern connected with Codeigniter components and their specific micro-tasks.



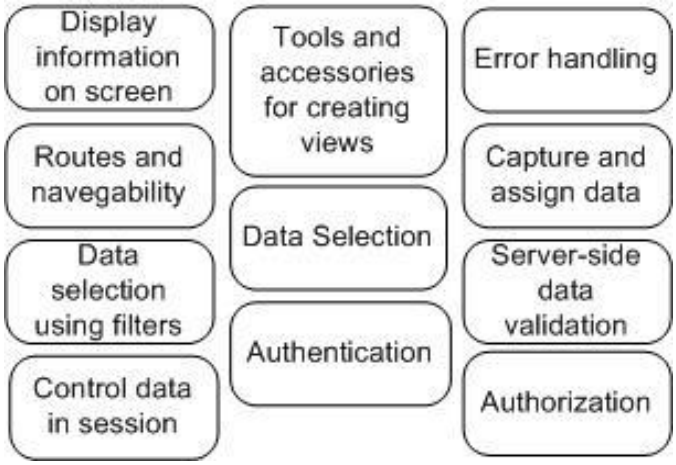
Selecting concerns examples

Suppose that a developer is requested to build an application module by using a new WAF. After the requirements elicitation process, the following requirements list is presented:

- The application has to extract the real estate information from the main database.
- Only admin users—already created in the database—can access the real estate information. Then, a login system is required.
- Admin can filter real estate information ordered by name, location or type.

We assume the requested developer should select the concerns listed in Figure 3.6. Similar to Figure 3.5, each concern of Figure 3.6 will be connected to its related components and tasks –as specified in table 3.4–. Concerns of the Figure 3.6 support developers as personalized learning guides, i.e., before starting the learning process, developers can discard some documentation unrelated to his/her needs.

Figure 3-6: Example of concerns selection.



3.2.5 Web application concerns summary

Nowadays, separation of concerns has been applied to different phases of the development process. This concept essentially means that dealing with complex problems is only possible by dividing the complexity into sub-problems. WAF learning could be considered as a complex problem, so we decided to use this concept to face this problem. We analyzed some web applications projects in order to find the common developers concerns. This analysis showed that no matter how different seems each application from one another, they use similar concerns. After this process we developed a new web application concern list. The main idea is developers have different requirements which mean they have different learning interests or concerns. So, if at the begging they identify what are their specific related concerns, they could find the specific WAF material that is related with their requirements. Finally, we connected web application concerns list with WAF components list and their micro-tasks. This connection gives the possibility to know for each concern what are the specific components and micro-tasks related to start the personalized learning process.

3.3 Introducing the use of examples

3.3.1 Introduction

This section deals with the last two issues: “good documentation is difficult to find and is often outdated” and “WAF documentation material is limited”. We initiate the section by introducing the importance of good examples. We establish some benefices of their incorporation into the new WAF learning technique; finally, we present a list examples connected with the previous concerns.

3.3.2 The use of examples

Related work

Some authors have emphasized the evolutionary importance of learning by observing and/or imitating what other people do, say, or write. Some of them agree it would be impossible (not to mention quite dangerous) for a human being to discover by one's own experience the vast amounts of knowledge that our ancestors developed over thousands of years. It is much more efficient to borrow this knowledge from others and reorganize it to fit in with one's existing knowledge and use it to one's own purposes [33].

Research on studying worked examples has consistently shown that for novice learners, instruction that relies more heavily on studying worked examples than on problem solving is more effective for learning, as well as more efficient in that better learning outcomes are often reached with less investment of time and effort during acquisition [34][35].

In the software area, besides of being useful for learning, providing examples have some others positive benefits:

- The use of examples can reduce the amount of typing required to complete a task, or ensure that the details of the code are correct [36].
- Finding existing applications that match requirements, and subsequently can be used as prototypes, would reduce the cost of many software projects [37].

However, it is difficult for developers to find appropriate code examples [37][45]. Sometimes they find wrong or deprecated material and solutions. Besides, examples

alone are not sufficient for the effective learning of application frameworks. First, although examples may contain concrete solutions that are useful to developers, they do not explicitly explain how the demonstrated solutions are provided by the framework [7].

A real scenario of the use of examples

The primary goal of web developers is to deliver high-quality software efficiently and in the least amount of time whenever possible [43]. But as we see in the previous chapters, the learning process is difficult and time-consuming. The different framework documentation techniques require that the developer puts much effort to browse and find the good documentation [44].

Let's check this real scenario:

A person asked this question "how to pass id in controller from form action using Codeigniter" in StackOverFlow website [46]. These kind of websites are plenty of good people who try to answer others questions. So, there is another person who answered with the next code:

```
<form class="addinvestmentform" action="<?php echo
base_url();?>index.php/ctl_dbcont/input_investment" name="application"
method="post" >
<input type="hidden" name="my_id" value="<?php echo $id; ?>"/>
</form>
```

The previous answer was checked as the useful answer.

Now, imagine that there is a developer called "Juan" who has a similar requirement. By using Google, Juan has a probability of finding the previous link. And he could use that example as a base for his projects.

However, the previous code doesn't have good quality. The official documentation recommends using:

```
echo form_open('ctl_dbcont/input_investment ');
```

Instead of:

```
<form class="addinvestmentform" action="<?php echo
base_url();?>index.php/ctl_dbcont/input_investment" name="application"
method="post" >
```

The reason is simple: the first code will automatically insert a hidden CSRF field in your forms (that protects you application against CSRF hacking [47]). The second one doesn't have this protection.

This is not fault of the person who answered the question. Because, these specific details are not known by all developers, but this case shows that is difficult to know if an example (that doesn't belong to the WAF official website) is of good quality or not.

There are other real scenarios like finding wrong solutions or deprecated solutions which could affect the software quality.

3.3.3 Creating a list of examples

Commonly examples are transversal to the WAF architecture, a simple "hello world" example could cross through different WAF layers and components. These examples are useful to identify the relationship between different components. Due to examples characteristics and based on the previous statements, we decided to create a list of examples that we consider useful for WAF learning. These examples are connected to web application concerns (described at section 3.2). Each concern has associated an example, gluing together the components and micro-tasks. As a bonus, these exercises provide a source of code reuse.

Table 3.5 exhibits the list of the proposed examples. Each concern has associated one example. Even when the example description is the same for all WAFs, each example has its own solution in a specific WAF –due to the different WAFs syntax and their own elements–. Similar to micro-tasks, these examples must be codified by senior WAF developers. Table 3.5 shows:

- Concern: the name of the concern involved.
- Description: a brief description of the example.
- Elements to be coded: a list of elements that a senior WAF developer has to implement in order to develop a functional example in a specific WAF.
- Name: the name of the mini-application or example.
- Base: the name of the base example. Some examples use a previous example as a base and redefined it.

Conventions:

- [C:test] indicates to create a controller with name ‘test’.
- [V:test] indicates to create a view with name ‘test’.
- [M:test] indicates to create a model with name ‘test’.
- [F:test] indicates to create a file with name ‘test’.
- [C:test_2] <- [C:test] indicates to copy the content of controller ‘test’ into controller ‘test_2’.

Table 3-5: List of examples of each concern.

Concern	Description	Elements to be coded	Name/Base
Display information on screen	This is a hello world example that is composed by 3 views and in the second view shows a message "welcome to the app"	• [C:hello] • [V:header] (Design it as you consider). • [V:hello] Put a message "welcome to the app". • [V:footer] (Design it as you consider).	**Name:** hello_world
Stylized screens	In this example the senior WAF developer suggest some style modifications to a 3 simple views.	[C:hello_2] <- [C:hello] [V:header_2] <- [V: header] [V:hello_2] <- [V:hello] [V:footer_2] <- [V: footer] • [F:bootstrap.css] Implement bootstrap CSS (or the default WAF CSS). • Decorate [V:header_2] [V:hello_2] [V:footer_2] (Design it as you consider).	**Name:** hello_2 **Base:** hello_world
Tools and accessories for creating views	This example explains how to create the views using the proper WAF elements. In this case how to show a list of people and a form to add new ones (the form isn’t functional).	• [C:people] Create a list of 5 people (with just a name and an email). • [V:people] Display the list of persons (using WAF elements to create tables) and create a form to add new people (using WAF elements to create forms). The form doesn’t have to be functional.	**Name:** people
Routes and navigability	This example explains how to navigate between different app sections. In this case between home and about sections.	[C:routes] <- [C:hello] [V:routes] <- [V:hello] • [V:routes] Create an additional text "About Section" and implement a link in this text to 'about section'. • [V:about] Display short information about the app developer (as you	**Name:** routes **Base:** hello_world

Concern	Description	Elements to be coded	Name/Base
		want).	
Capture and data assignment	This example uses a simple currency converter in order to explain how to capture and assign data. The user enters an amount and rate and the app simply executes a multiplication.	• [V:currency] Create 2 inputs and 2 texts (amount – rate; “Enter the amount”, “Enter the rate” respectively) and a submit button ‘Send’. • [C:currency] Capture this information by post in the proper method and assign the result to the [V:currency_res]. • [V:currency_res] show the result of multiply the amount by rate. • Try to use a method to prevent: SQL injection and CSRF and XSS attacks.	**Name:** currency
Client-side data validation	Based on the currency converter, this example adds a client-side validation.	[C:currency_client] <- [C:currency] [V:currency_client] <- [V:currency] [V:currency_client_res] <- [V:currency_res] • [V:currency_client] Add client-side validation and validate that amount and rate inputs are required, are numeric values and aren’t negative numbers.	**Name:** currency_client **Base:** currency
Upload files	This example explains how to upload PDF files to the app through a simple form.	• [V:file] Create an input type file and a submit button ‘Send’. • [C:file] Validate the file type (only PDF files are allowed). Later upload this file to a folder called uploads (or the proper WAF uploads folder).	**Name:** file
Error handling	Based on the currency converter with the server validation ‘currency_handling’, this example adds a proper way to catch the errors and display them.	[C:currency_handling] <- [C:currency_server] [V:currency_handling] <- [V:currency_server] [V:currency_handling_res] <- [V:currency_server_res] • [C:currency_handling] Eliminate the alert ‘Error’. In the case of one controller validation fails over capture the error and show the error in [V:message] (or use the proper WAF view error mechanism) also shows [V:currency_handling] repopulating the previous data (amount - rate).	**Name:** currency_handling **Base:** currency_server
Internationalization	This example explains how to manage different language packages. If a user clicks	• [C:international] • [V:international] Display 2 hyperlink texts (‘English’ and ‘Spanish’) each hyperlink call a method of [C:international] and send different vars by GET (‘en’ – ‘spa’,	**Name:** international

Concern	Description	Elements to be coded	Name/Base
	'English' it will be displayed a message: "Welcome this is an English text". And it the user clicks 'Spanish' it will be displayed a message: "Bienvenido este es un texto en español".	respectively). • [V:international_text] Depending on each var previously selected, [C:international] will be load a different language file (that contains vars with texts) and it will display [V:international_text] with the proper vars (NOT static text) with the texts ("Welcome this an English text", "Bienvenido este es un texto en español", respectively).	
Localization	This example explains how to manage the users' locations. It displays different information depending of the user location.	• [C:localization] Capture the location where the user is. • [V:localization] Display a message: "You're in US" (if the user is in US) or "You're outside US" (if the user is outside US).	**Name:** localization
Caching	Based on user selection, this example explains how to read a list of users directly from a cache file instead of a database.	[M:user] <- [M:user] [V:cache] <- [V:user_select] [C:cache] <- [C:user_select] • Create all items from notes 1 and 2*. • [C:cache] extract all users from the database and show them in [V:cache]. But, if the information is cached, discard read information from the database and read it directly from a cache file.	**Name:** Cache **Base:** user_select
Testing	Based on user selection, this example explains how to active the debug system and how to display different information like: queries, memory use, get and post data, among others.	[M:user] <- [M:user] [V:debug] <- [V:user_select] [C:debug] <- [C:user_select] • Create all items from notes 1 and 2*. • [C:debug] extract all users from the database and show them in [V:debug]. Enable the debug system, show the content of the variables used, the queries, and the memory used.	**Name:** test **Base:** user_select
Portability	This example explains how to display different views depending of the user's device (for example if a user is navigating using a desktop computer or a	• [C:portability] Detect if the user is navigating in a desktop computer or a tablet. If the user is navigating in a desktop it'll display [V:desktop_index] that is inside subfolder "page" and if the user is coming from a tablet it'll display [V:tablet_index] that is inside subfolder "tablet". • Create two subfolders "page",	**Name:** portability

Concern	Description	Elements to be coded	Name/Base
	tablet).	"tablet" (inside views root folder). • [V:desktop_index] Display a message "Navigation from desktop". • [V:tablet_index] Display a message "Navigation from tablet".	
Data Selection	This example extracts all users from the database and displays them.	• Create all items from notes 1 and 2*. • [M:user] • [C:user_select] extract all users from the database and show them in [V:user_select].	**Name:** user_select
Data Selection with pagination	Based on user selection, this example displays the users' information by pages. 5 users by each page.	[M:user] <- [M:user] [V:user_select_pags] <- [V:user_select] [C:user_select_pags] <- [C:user_select] • Create all items from notes 1 and 2*. • Create at least 7 new users using the database manager. • [C:user_select_pags] extract the first 5 users from the database and show them in [V:user_select_pags]. By default the page app will be 1. So, the controller has to capture the page and depending of the page display the proper users.	**Name:** user_select_pags **Base:** user_select
Data selection usign filters	Based on the user selection this example shows how to extract all the 'female' users from the database and display them.	[M:user] <- [M:user] [V:user_select_filter] <- [V:user_select] [C:user_select_filter] <- [C:user_select] • Create all items from notes 1 and 2*. • [C:user_select_filter] extract all the female users from the database and show them in [V:user_select_pags].	**Name:** user_select_filter **Base:** user_select
Multiple data selection	This example explains how to select data from multiples models. It extracts all information from users and their respective quotations.	• Create all items from notes 1, 2, 3 and 4*. • [M:user] • [M:quotation] • [C:multiple] extract all users with their specific quotations from the database and show them in [V:multiple].	**Name:** multiple
Data storage	This example explains how to save data (a user) at the database.	• Create all items from notes 1 and 2*. • [C:user_storage] • [V:user_storage] that display a form with input texts: name and email. And select gender ('male', 'female'). After the a user completes and sends the form, the app will assign the data to [M:user] or/and use a model function and it'll add the data to the database.	**Name:** user_storage

Concern	Description	Elements to be coded	Name/Base
Data editing	This example explains how to edit data. It allows entering a user id and after that allows editing the user selected. Finally it updates the user's information at the database.	• Create all items from notes 1 and 2*. • [M:user] • [V:choose_user] Display an input text to collect the user 'id' and a submit button 'Send'. • [C:user_update] When the submit button is pressed this controller displays another view [V:user_update] that shows the complete information of the user previously selected. And allows editing it. • [V:message] Display a message 'User successfully updated' when the user is edited.	**Name:** user_update
Deleting Data	This example shows how to delete a user from the database.	• Create all items from notes 1 and 2. • [M:user] • [V:user_delete] Display an input text to collect the user 'id' and a submit button 'Send'. When the submit button is pressed the user has to be deleted from the database and a success [V:message] is displayed.	**Name:** user_delete
Creating model functions	This example explains how to create model functions. In this case it's created a function 'gender' which returns the name of the female users.	• Create all items from notes 1 and 2*. • [M:user] Create a function called 'gender' that receives all users' info and return the name of all female persons. • [C:gender] Collect all users' info. Send this info to [M:user] and display the female names at [V:gender].	**Name:** gender
Model-side data validation	Based on the user storage, this example explains how to manage the model-side validation. How to validate if an element is required or is an email, among others.	[M:user] <- [M:user] [V:user_storage_validate] <- [V:user_storage] [C:user_storage_validate] <- [C:user_storage] • Create all items from notes 1 and 2*. • [M:user] Create the proper functions or methods to validate that name is required, email is a proper email and gender is 'male' or 'female' (implement this methods in the controller if is suggested by the WAF). • [C:user_storage_validate] After the form is send, if the validation fails, the app will display a message corresponding to the error in [V:message]. After, it will display the form [V:user_storage_validate] repopulating with the previous data.	**Name:** user_storage_validate **Base:** user_storage

Concern	Description	Elements to be coded	Name/Base
		If the validation is correct the app will assign the data to [M:user] or/and use a model function and it'll save the data to the database.	
Authentication	This example shows a simple login system. Using an email and a password.	• Create all items from notes 1 and 2. • [V:login] Create a form with 2 inputs 'email' and 'password'. • [C:login] Collect the form information and compared with the user table (also it should be storage in session). Finally it'll display a message [V:message] logged in completed. • Try to use an encryption method for the password.	**Name:** login
Authorization	This example shows a message depending if a user is logged or not.	• [C:admin] If a user is logged and try to access to this controller, it will display a message [V:message] "welcome to the admin section", else, it will display "unauthorized access".	**Name:** admin
Control data in session	This example explains how to add and control data in session. In this case how to add and show products to a simple cart.	• [V:cart] Display two products: 'product 1' – 'product 2' (only the name). Next to each product, display a button "Add to cart". • [C:cart] When 'Add to cart' is clicked the product is added to session. The products at session will be displayed at the bottom of the view (if there aren't products, it will show a text 'there aren't products in the cart').	**Name:** cart
Controller-side data validation	Based on the currency converter, this example adds a server-side validation. If a validation fails over it displays a simple 'error' message.	[C:currency_server] <- [C:currency] [V:currency_server] <- [V:currency] [V:currency_ server_res] <- [V:currency_res] • [C:currency_server] Apply some validations: amount and rate inputs are required, are numeric values and aren't negative numbers. These validations inside the controller. In the case of one controller validation fails over display an alert with the message 'Error'.	**Name:** currency_handlin g **Base:** currency_server
Coupling modules	This example explains how to connect a PDF module to the application. Allowing converting a text to PDF.	• [V:pdf] Create a textarea "description" and a submit button 'Send'. • [C: pdf] look at WAF website or forum for a PDF module or plugin that allows converting an html text to PDF. So, it will convert the "description" text to PDF file and finally the app will allow downloading	**Name:** pdf

Concern	Description	Elements to be coded	Name/Base
		the PDF file.	
Creating modules	This example shows how to create a new XLS module and how to convert data into XLS files.	• [F:xls] Create a module xls that allows converting arrays of data into XLS files. • [C:xls] Create 10 users. Using the xls module convert these users into a XLS file.	**Name:** xls
Auto-generated code	This example shows how to create a complete CRUD of users using the WAF tools.	• Create an example that shows how to create a CRUD of client [M:user] using the WAF tools.	**Name:** crud

***Notes:**

1. Create a database **test**, inside create a table **user** with the columns **id** (int - autoincrement - primary), **name** (varchar 100, not null), **email** (varchar 100, not null), **gender** (varchar 10, not null) and **password** (varchar 100, not null).
   Create a model [M:user] with the previous information if it's required.
2. Using your database manager (like phpmyadmin) create two users: ('1', 'Daniel', 'dan@test.com', 'male', 'test567') - ('2', 'Sara', 'sara@test.com', 'female', 'test568').
3. Create a table **quotation** with the columns **id** (int - autoincrement - primary), **user** (int, not null), **date_created** (date, not null), **description** (varchar 10000, not null), total (varchar 100, not null).
4. Using your database manager (like phpmyadmin) create two quotation ('1', '1', '2014-03-03', 'test test', '20 USD') - ('2', '2', '2014-03-05', 'test test test', '100 USD').

3.3.4 Use of examples summary

The use of examples have the following benefits: (i) they can reduce the amount of typing required to complete a task, (ii) finding existing examples that match requirements, can serve as a base of code-reuse, and (iii) they can portraits the WAF architecture and help to understand how the WAF components and how the elements are connected. Based on these statements we decided to incorporate examples in the new WAF learning technique. Due to examples characteristics, we created a list of examples associated with the previously defined concerns. At this point we have a list concerns which are connected with WAF components and micro-tasks. Besides, is proposed a list of examples for each concern. As micro-tasks associations, we suggest the example codification process should be done by a senior WAF developer.

3.4 A new WAF learning technique

3.4.1 Introduction

In the previous sections of this chapter we have presented some crucial elements to improve WAF learning: (i) at section 3.1, we analyzed different WAFs and we identified WAFs common components and their associated micro-learning-tasks or micro-tasks. With these elements, we have a complete panorama of the different elements that developers may to learn to develop web applications with WAFs; (ii) at section 3.2, we identified a list of common web application concerns, together with a connection of WAF components and micro-tasks; and (iii) at section 3.3, we propose to improve the learning material by defining a set specific examples (to be coded on each WAF) for each concern, as companions of the related micro-learning-task.

Based on those concepts we define the learning steps in the new WAF learning technique: (i) in the first step the learner extract his application requirements and select the web application concerns related with his/her requirements; (ii) the corresponding micro-tasks documentation to each component related with each concern is presented to the learner, the learner has to read and follow this documentation in order to acquire WAF knowledge and understand the WAF components; (iii) micro-learning-tasks are, however not enough to acquire application level skills. Then, parallel to this, for each concern a meso-task –example– documentation is also presented, the learner has to read the micro-tasks and codify the meso-tasks in order to obtain more knowledge; and (iv) only after the learner has deal with the micro-tasks and meso-tasks, he/she is ready to confront the development of his/her own application –the macro-task–.

At the end of this section, we present the design of a web application to support the new WAF learning technique. The main objectives of this application are: (i) provide a mechanism to complete the micro and meso tasks –these should be completed by senior WAF developers–, (ii) facilitate the access the learning material, (iii) establish a mechanism to allow learners to select their concerns and present the specific learning material to each of them.

3.4.2 Micro-learning and the definition of the new WAF learning technique

No matter how learning is conceptualized, in all cases there is the possibility of considering it in terms of micro, meso and macro aspects, levels or tasks [20]. This is the main concept of Micro-learning. This division theory can be applied to all kind of areas.

Bruck et al [48] implemented micro-learning at the University of Innsbruck within a class of philosophy for science. In this class, they developed an application of learning cards and they integrated it in the PC screensavers. Also, they implemented a similar technology in a governmental entity of the Republic of Austria. The goal of the implementation was to drive training of public servants and thus help to improve the quality of governmental services provided to the public. Watson [49] uses micro-learning to separate and create different learning objects for different courses. Kovachev et al [50] use micro-learning in a case of bilingual vocabulary learning.

Taking advantage of micro-learning characteristics and the new WAF learning elements defined, we decided to separate the new WAF learning into micro, meso and macro tasks. In this case, the micro-tasks correspond to the micro-tasks defined at section 3.1.3, the meso-tasks correspond to the examples defined at section 3.3.3 and the macro-task corresponds to the web application that the learner has to develop.

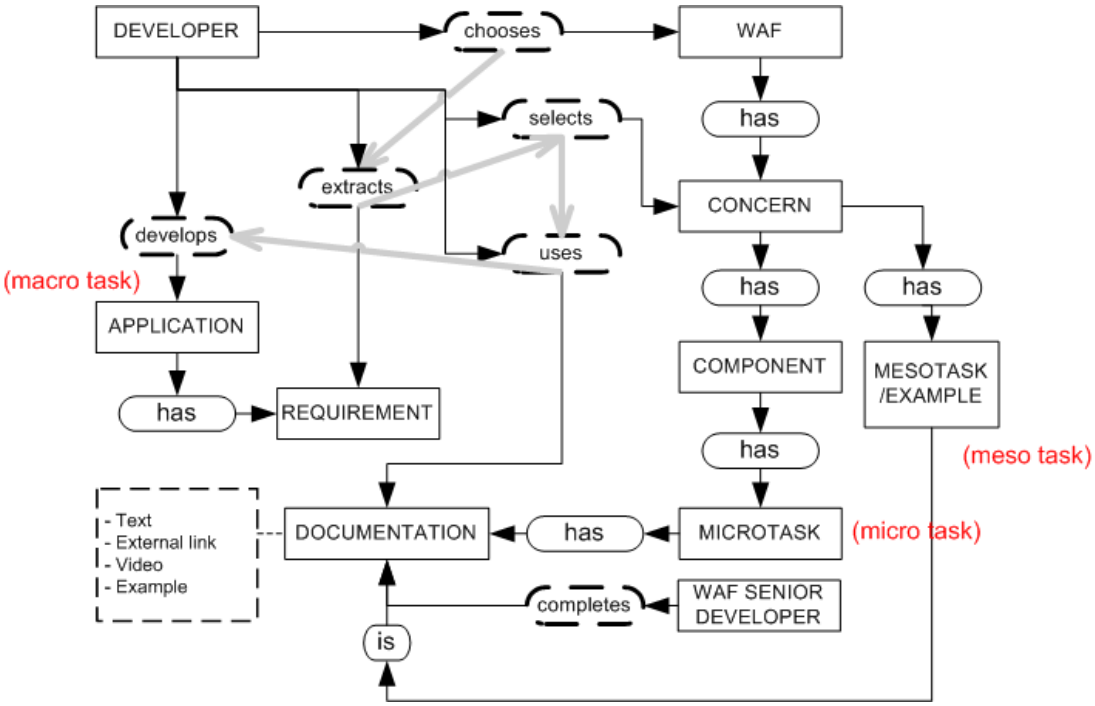
Commonly these micro-tasks could be developed in just a few of second or minutes; to macro-tasks that could be developed in hours, days or months [20].

Figure 3.7 shows a real example of micro, meso and macro tasks division over Codeigniter WAF for the proposed WAF learning.

In this figure, some texts are in Spanish. It's because we use some students whose native language is Spanish, for developing the laboratory case that is described at chapter 4. It's important to highlight that micro and meso tasks must be completed by a senior WAF developer (later, we propose an application to complete these tasks). Macro-tasks are global exercises application or simply the application that the WAF learner has to develop.

Figure 3-7: An example of Micro, meso and macro tasks over Codeigniter.

Figure 3-8: A proposed representation of the new WAF learning environment.



In Figure 3.8, we summarize the new WAF learning technique process. Developer first step is to choose the specific WAF in which he/she wants to develop the application. The second step is analyzing the application to develop and extract the requirements. Third, he/she has to choose the concerns related to the application that support the previously

requirements. Finally, he/she has to work with the specific elements: examples and documentation tasks (previously filled by a senior WAF developer) in order to build the application. In this case the micro tasks are the components micro-tasks, the meso tasks are the concerns examples and finally the macro tasks is the application to be developed.

3.4.3 DL Application

Tasks documentation is a Senior WAF developer task. After this documentation is completed, the WAF learner has to access to this information in order to complete the application he/she has to develop, and to learn the WAF characteristics. In order to improve these activities we developed a web application called “DL application” or driving-learning application. This application not only serves as a mechanism to complete these tasks and display this information, but also, it gives a mechanism to drive the WAF learning. Figure 3.9, 3.10 and 3.11 shows a real scenario using DL application. These figures detail each step that a WAF learner has to take in order to develop a web application over Codeigniter.

Figure 3.9 shows the home page of DL application [51], in this view the WAF learner has to select the WAF he/she wants to work with, and after he/she has to select the concerns related with his/her needs (this figure shows information in Spanish because we developed some material to be use by people native in this language). At the end this figure shows 2 options “Get examples” to get the proper meso tasks, and “Get Documentation” to get the proper micro-tasks.

Figure 3-9: Home page of DL application.

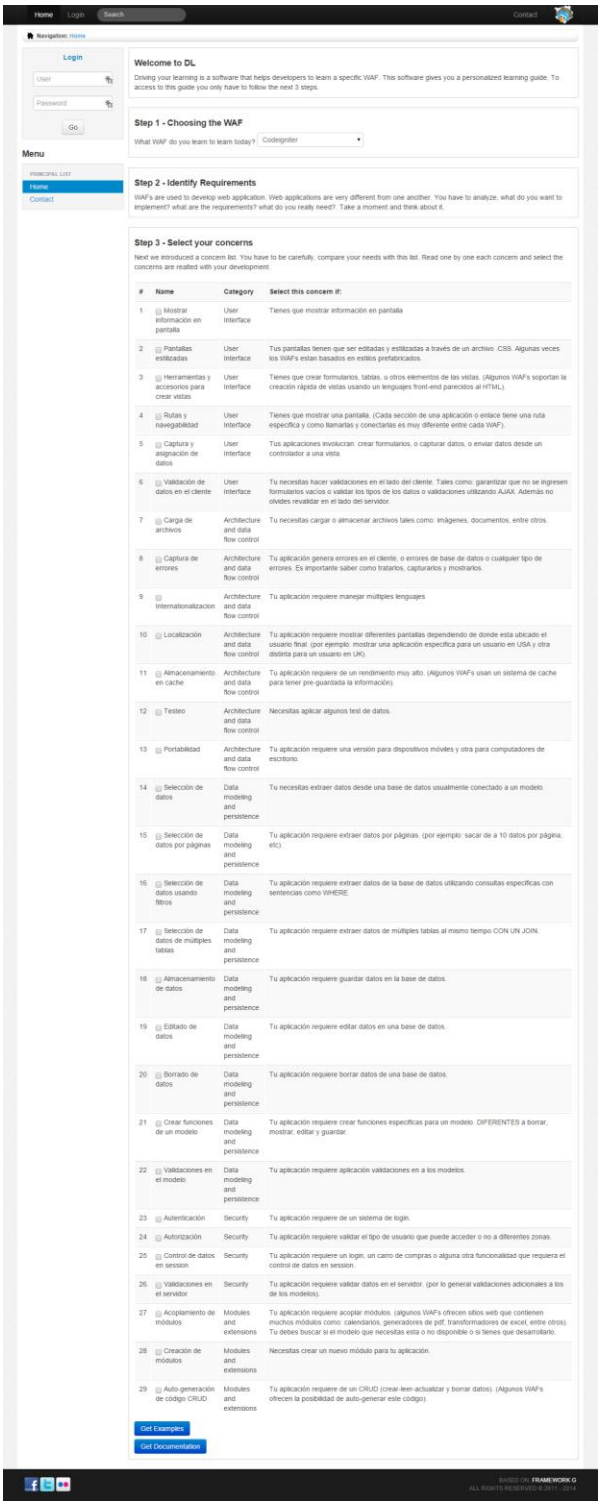
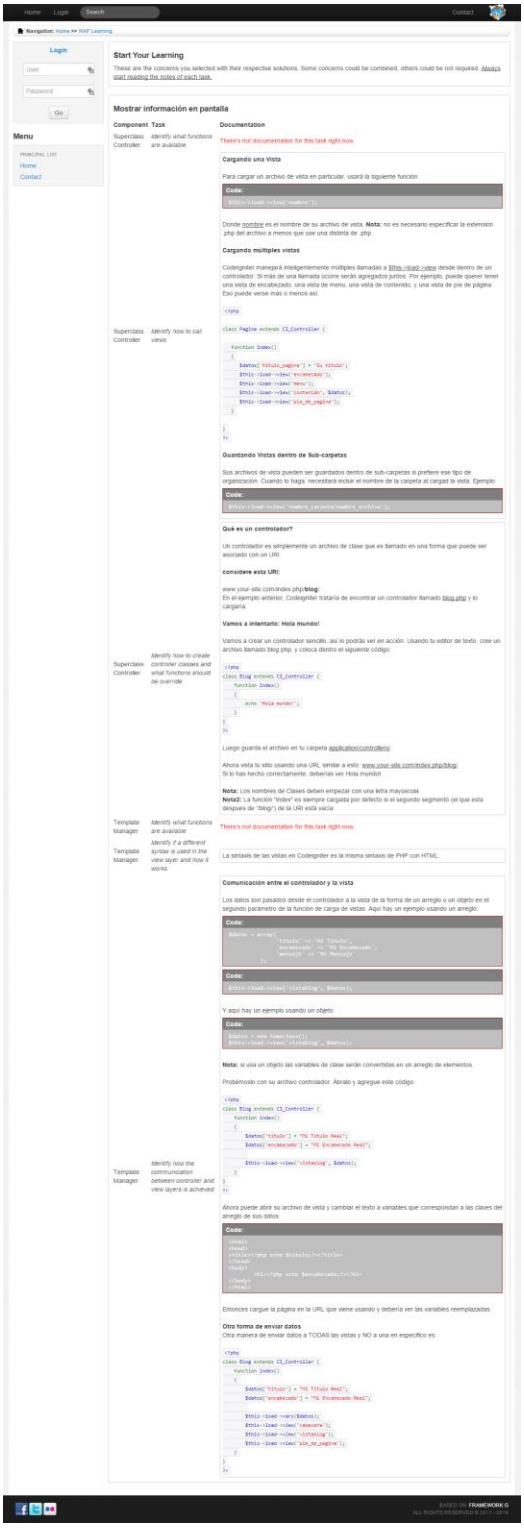
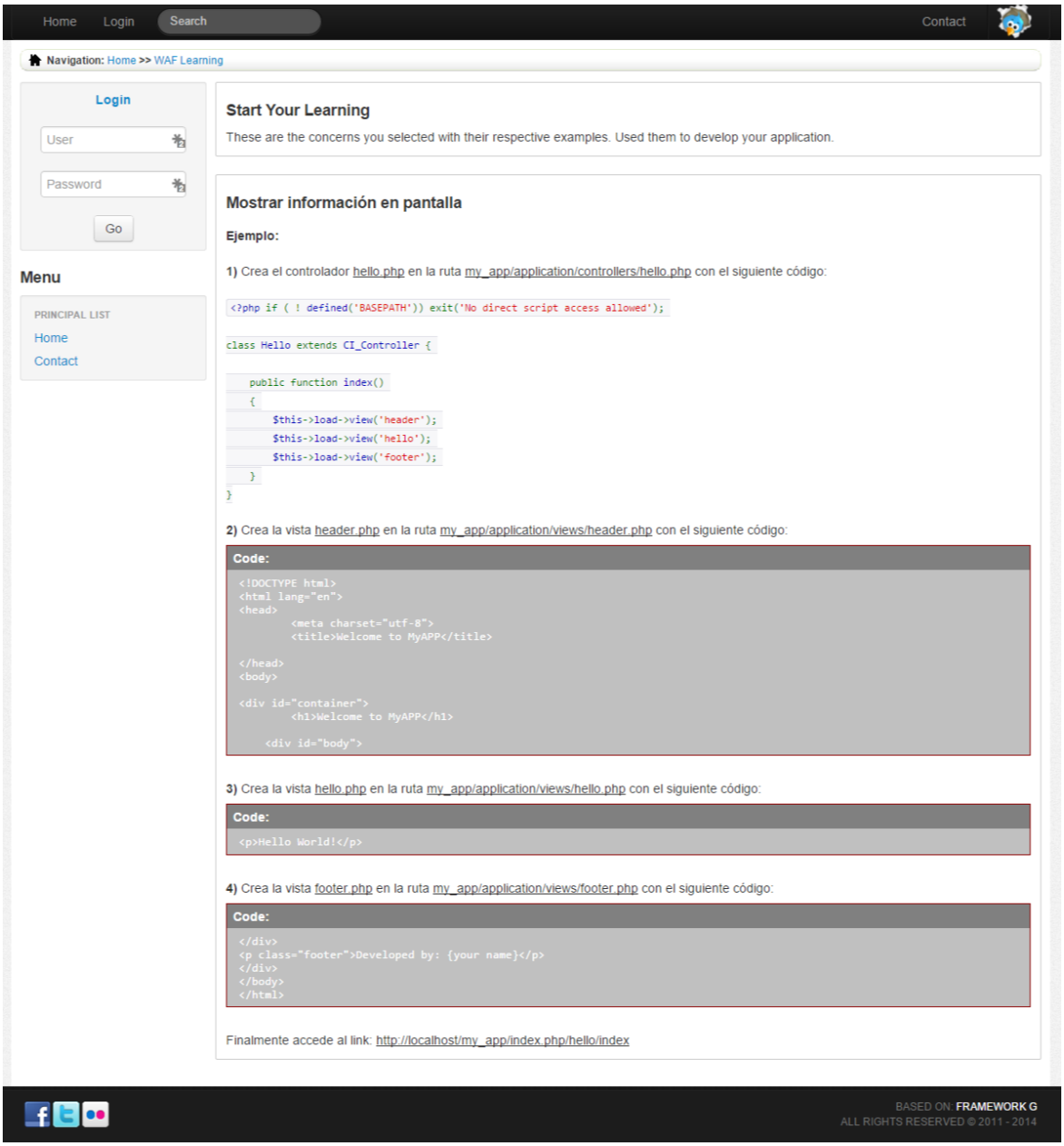


Figure 3-10: Micro-tasks documentation view over DL application.



Spouse that the WAF learner only selected the first concern “Display information on screen”, because, he/she only needs to develop an under construction web page. After he selected this concern, he/she clicked over “Get documentation” button. Then, the DL application will display the micro-tasks associated with this concern. Figure 3.10 shows the proper documentation displayed to the WAF learner. The learner will use this material to better understand the different WAF elements involved in this concern.

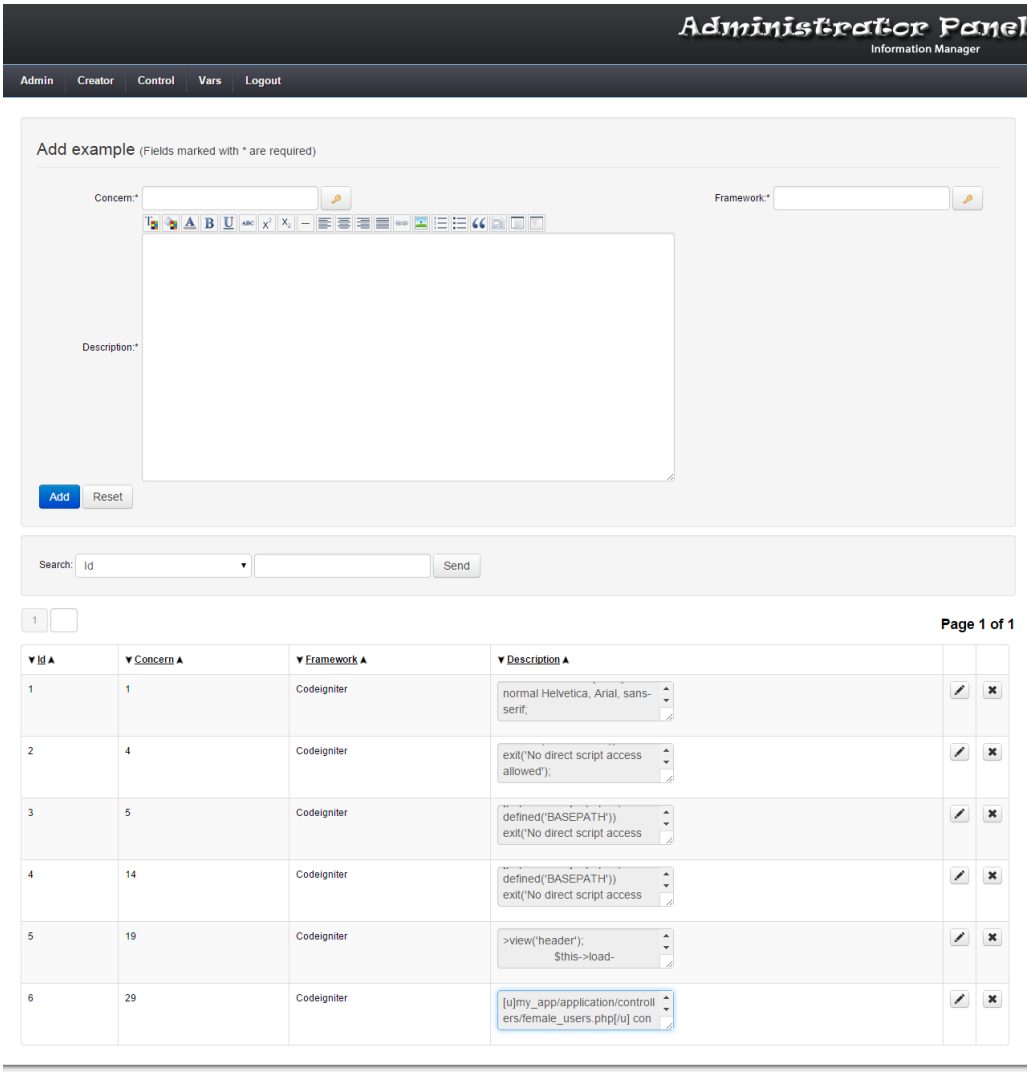
Figure 3-11: Meso-tasks view over DL application.



Finally, figure 3.11 shows the meso-task (or example) associated to the concern “Display information on screen”. The main idea is the learner use those elements (micro and meso tasks) to develop the required under construction web page.

The previous three figures show how the WAF learners drive their own learning. But also, WAF senior developers require different elements to complete the WAF learning material. Figure 3-12 shows the admin panel of the “DL Application”, inside this panel the WAF senior developers are able to complete the micro and meso tasks for different WAFs. Notice they require to login to access to this zone.

Figure 3-12: DL application admin panel.



3.4.4 A new WAF learning technique summary

This chapter defines the new WAF learning technique. By using the previous element from the previous sections, we divided the learning technique into micro, meso and macro tasks. The micro-tasks defined as the components micro-tasks; the meso-tasks defined as the concern examples; and the macro-task defined as the application that the WAF learner has to develop. We proposed a representation of the new WAF learning environment which shows the new learning path.

Finally, this section describes a new web application called “DL application” that allows WAF senior developers to complete the different material and allows the WAF learner to drive his/her learning. Besides, some figures of the real application are showed to provide a real panorama performance of the learning application.

4. Laboratory case

4.1 Introduction

This chapter describes a quasi-experiment. This quasi-experiment is developed in order to better-understand and to analyze the new WAF learning technique. The idea is to obtain information about different aspects like:

- How learners adopt the new technique.
- How this technique improves –or not– the WAF learning.

This study was intended to provide evidence that the new WAF learning technique could improve and drive the developers WAF learning. The principal hypothesis is that a novice developer could save time by using the new WAF learning technique to develop an application versus the common WAF learning techniques, specifically cookbooks or official web tutorials. This experiment took groups of similar undergraduate students and put them within a controller experimental environmental, they had to develop a set of tasks by using a new web application framework –in this case Codeigniter–. The final results support the hypothesis: a novice WAF developer saves time by using the new WAF learning technique to develop an application in a new WAF.

To develop this laboratory case, we employed some aspects from Flores' Thesis [19] specifically of his quasi-experiment description and design. He also used some groups of students to analyze a framework learning environment. We adapted his quasi-experiment to our own requirements.

4.2 Academic Quasi-Experiment

The use of empirical studies with students (ESWS) in software engineering helps researchers gain insight into new or existing techniques and methods. However, due mainly to concerns of external validity, these studies are often viewed skeptically by

researchers and practitioners. Empirical studies with professionals, which are widely accepted by the above-mentioned also suffer from similar generalizability problems. Therefore, just like any other empirical studies, ESWSs can be valuable to the industrial and research communities if they are conducted in an adequate way, address appropriate goals, do not overstate the generalizability of the results and take into account threats to internal and external validity [52]. As ESWSs are often used to obtain preliminary evidence in support of or against research hypothesis, this experiment was designed as such. The independent experimental validation of claims is not as common in Software Engineering as in other, more matured sciences. As such, the quasi-experiment here detailed was designed to be performed in different locations, and by different researchers, in order to enhance the ability to integrate the results obtained and allow further meta-analysis on them.

4.2.1 Participants selection

The experiment subjects were 15 undergraduate students from System and informatics engineering at the National University of Colombia –Sede Medellín–. They were part of the eighth semester of this career, and they attended to an optional course on “Design and construction of software products”. This course was about taking some previously worked projects, and design and implement them into a real software product using php language with simple code divisions –like MVC pattern– but without the use of WAFs. We only took two classes of this course to elaborate the quasi-experiment. In the first class students had to complete a pre-questionnaire –which only took 15 minutes–, in the second class they had to develop the experiment –which took 2 hours–.

These subjects were divided into three groups; each group had its own characteristics.

- **Group 1 (G1) –or baseline–:** this group served as the control group. Its subjects used the WAF official tutorial or cookbook to develop the experiment.
- **Experimental Group 2 (G2):** this group used a DL application section [54] that allows only using meso-tasks or examples. Only a section of the new WAF learning material.
- **Experimental Group 3 (G3):** this group used a DL application section [54] that allows using micro-tasks and meso-tasks. The complete new WAF learning material.

4.2.2 Pre-experiment evaluation

For an experiment of this kind, it is important to assure that the subjects are similar and that their base skills don't pose a significant threat to the validity of the results. Therefore, they were scrutinized based on their academic track, by analyzing their grades on a selected subset of courses. These courses were deemed relevant to the outcome of the experiment, namely: (I) Programming Fundamentals, (II) Data Structures, (III) Programming Object Oriented, (IV) Software Engineering, (V) Databases I, (VI) Programming Logical and Functional, and (VII) Requirements Engineering. Their grades can be found in Appendix A, Table A.1. An independent samples t-test was conducted to compare the average students' grades (shown in Table 4.1) between the baseline (G1) and other experimental groups (G2, G3) –grades were between 0 and 5–.

As shown in Table 4.2, there was no significant difference in the scores for the Experimental Group 2 (M = 3.83, SD = 0.39) and Baseline Group 1 (M = 4.02, SD = 0.60) conditions; p = 0.573, within a 95% confidence interval.

As shown in Table 4.3, there was no significant difference in the scores for the Experimental Group 3 (M = 3.81, SD = 0.47) and Baseline Group 1 (M = 4.02, SD = 0.60) conditions; p = 0.557, within a 95% confidence interval.

Table 4-1: Student grades group statistics.

Group	N	Mean	Standard Deviation	Standard Error Mean
G1 - Baseline	5	4,027	0,603	0,2697
G2	5	3,837	0,399	0,1785
G3	5	3,817	0,475	0,2123

Table 4-2: Baseline G1 vs. Experimental Group 2 Independent Samples Test. The first column is the Levene's Test for Equality of Variances, showing a significance greater than 0.05 (0.221). The other three columns are the t-test for Equality of Means. Since we can assume equal variances, the 2-tailed value of 0.573 allows us to conclude that there is no statistically significant difference between the two conditions.

G1 vs G2	Sig Levene	T	DF	Sig. (2-tailed)
Eq. Var. Assumed	0,2219	0,5876	8	0,573
Eq. Var. Not Assumed		0,5876	6,9392	0,575

Table 4-3: Baseline G1 vs. Experimental Group 3 Independent Samples Test. The first column is the Levene's Test for Equality of Variances, showing a significance greater than 0.05 (0.837). The other three columns are the t-test for Equality of Means. Since we can assume equal variances, the 2-tailed value of 0.557 allows us to conclude that there is no statistically significant difference between the two conditions.

G1 vs G3	**Sig Levene**	**T**	**DF**	**Sig. (2-tailed)**
Eq. Var. Assumed	0,8377	0,6117	8	0,5577
Eq. Var. Not Assumed		0,6117	7,5841	0,5586

4.2.3 Framework selection

In order to select a WAF, we have to consider some aspects:

- Had to be unknown for all participants.
- The WAF official documentation had to be in Spanish (due to participants native language) and available online.
- Had to be open-source.
- Had to be known for the authors, in order to complete the documentation.

Due to the previous research we developed at chapter 3, we had knowledge working with six WAFs –Codeigniter, Yii, Prado, MVC4, Ruby on Rails and Cakephp–. We selected Codeigniter due to the experience we have, and because this WAF fulfills all the above requirements.

4.2.4 Pre-questionnaire

The first phase of the experiment was to hand out a questionnaire to the students. The questionnaires were designed using a Likert scale [41]. This psychometric bipolar scaling method contains a set of Likert items, or statements, which the respondent is asked to evaluate according to any kind of subjective or objective criteria, thus measuring either negative or positive response to the statement. For all the questionnaires in this experiment (both pre- and post-), the Likert items had a five-point format: (1) strongly disagree, (2) somewhat disagree, (3) neither agree nor disagree, (4) somewhat agree, and (5) strongly agree.

Pre-questionnaire content

The pre-experiment questionnaire was used to ascertain the students' background and general profile in order to screen out possible differences amongst the students regarding their basic skills. It also served to confirm the students' lack of acquaintance with Codeigniter. All groups were submitted to this questionnaire (see Appendix B), whose answers are detailed in Appendix C, and further analyses later in this chapter.

4.2.5 Pre-experiment requirements

To elaborate this experiment, we used a laboratory room at National University of Colombia –Sede Medellín– faculty of Minas. Due to Codeigniter requirements, we pre-installed in 15 computers three programs:

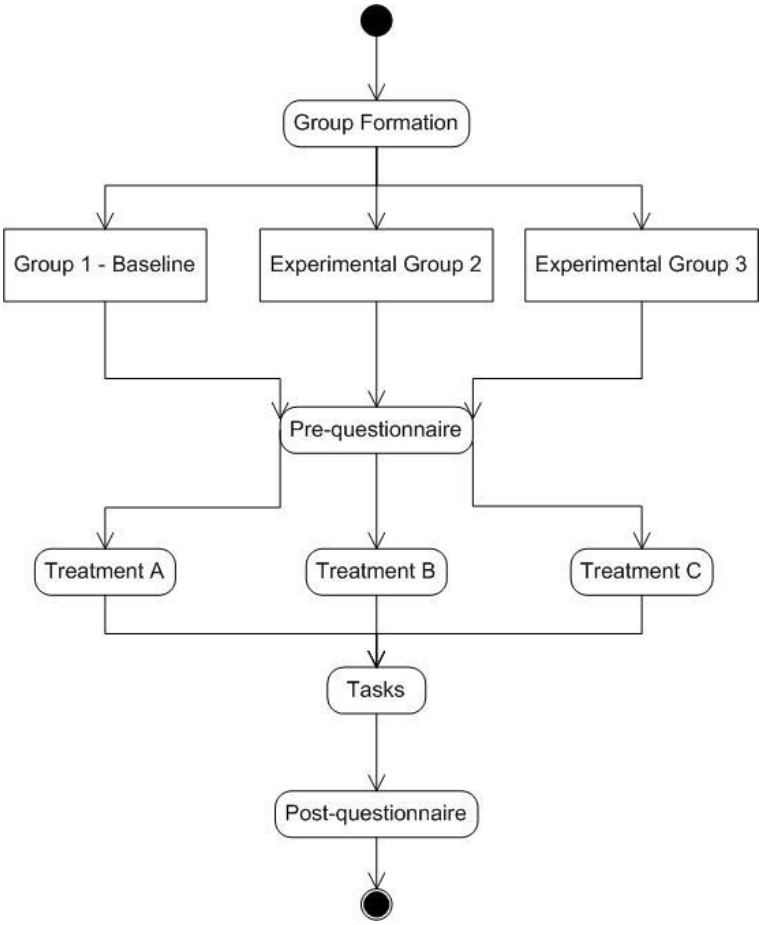
- WampServer 2.5 (a Windows web development environment) [53].
- Notepad++ 6.6.8 (a free source code editor) [54].
- Codeigniter 2.2.0 (an agile and open PHP web application framework) [55].

We moved Codeigniter folder to C:/wamp/www/ and rename the folder to my_app
c:/wamp/www/my_app/

4.2.6 Experiment description and treatments

This section describes the experiment phases and the group treatments. Figure 4.1 shows the experiment phases. The first phase was the group formation. Second, the students were submitted to a pre-questionnaire to establish their initial state. Third, each group was submitted to a different treatment and the students started to develop a set of tasks. Finally, the students were submitted to a post-questionnaire to obtain some results.

Figure 4-1: Experiment protocol and phases.



Experiment Setting

The experiment was conducted as in a laboratory classroom; we pre-installed some programs listed at section 4.2.5. The students had very limited internet access in order to minimize distractions (instant messaging, e-mail, etc.). Only the group 1 members had access to Codeigniter online documentation, and groups 2 and 3 had access to DL application.

Supervisors

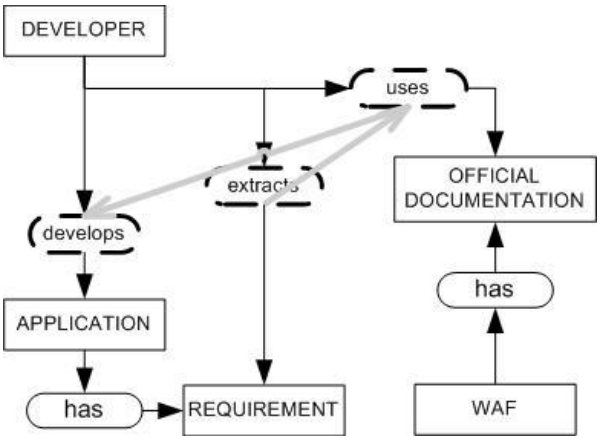
During the experiment, three colleagues helped us as supervisors. Their main task was to verify the students' tasks fulfillment. When a student finished a task, he/she raised his/her hand and a supervisor approached to him/her. The supervisor verified if the task was correctly fulfilled and finally took the time to complete each task (see tasks at section 4.2.7).

Treatments

After the pre-questionnaires phase, the students were submitted to a treatment phase, where each group was introduced to their own experiment environment.

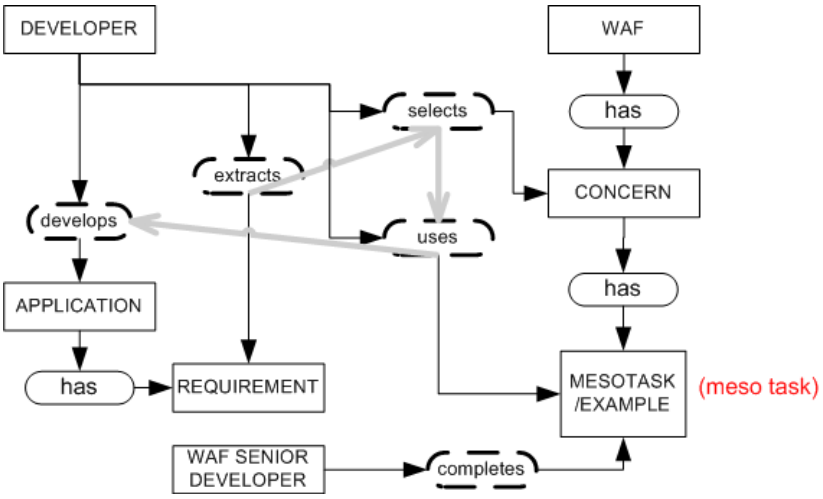
Treatment A: this treatment only applied for Group 1 –or baseline–. Figure 4.2 represents the learning path that the students had to follow to complete the tasks. They only used the Codeigniter Spanish official documentation [56].

Figure 4-2: Treatment A experiment environment.



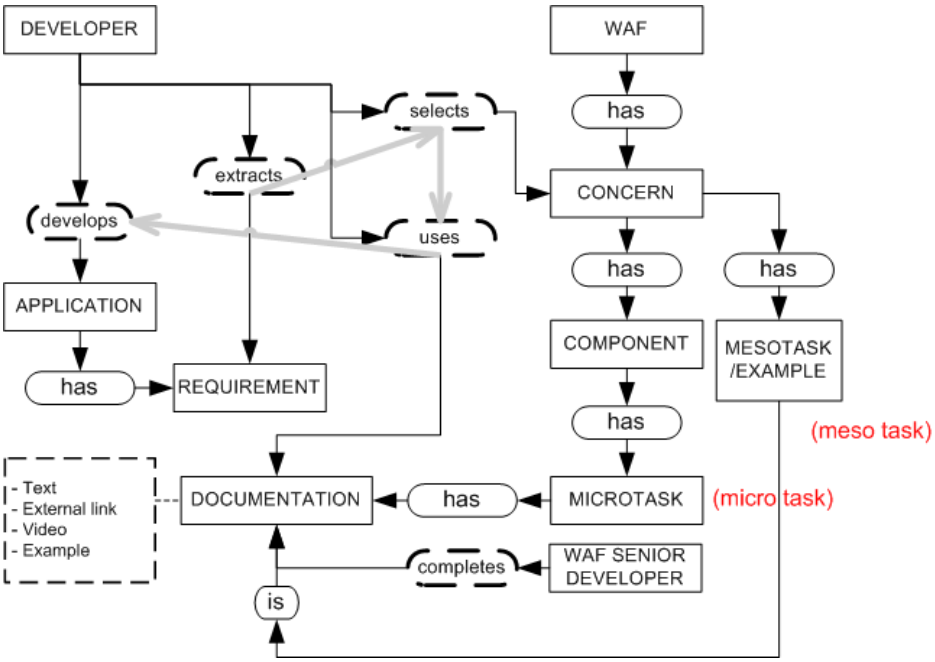
Treatment B: this treatment only applied for experimental Group 2. Figure 4.3 represents the learning path that the students had to follow to complete the tasks. They used the DL application but limited only to access to examples or meso-tasks [57].

Figure 4-3: Treatment B experiment environment.



Treatment C: this treatment only applied for experimental Group 3. Figure 4.4 represents the learning path that the students had to follow to complete the tasks. They used the complete DL application, accessing micro and meso-tasks information [58].

Figure 4-4: Treatment C experiment environment.



4.2.7 Macro-Task

At this point, the subjects were ready to develop the main part of the experiment. The macro-task was designed with the intention to develop a part of a big real application. This macro-task was divided into 4 iterations, the main idea was students could incrementally build this application, after each participant finished one iteration; a supervisor verified the task and took the time the participant spent on its development.

Appendix D shows the main document which was delivered to all subjects. In this document, the supervisor set the time when each subject completed each iteration.

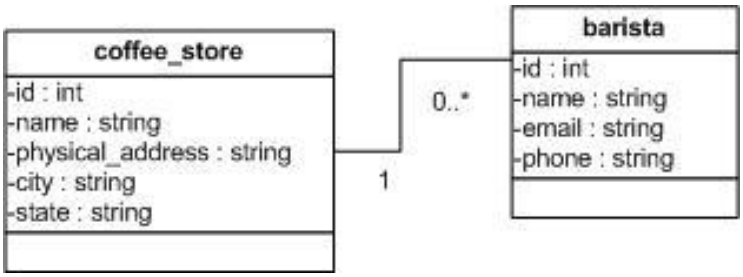
The following text was presented to the students and was the introduction to the experiment:

> *A software company required your services to develop an application to manage information of different coffee stores around United States. After the software requirements elicitation process, the company established the next features:*

Due to the quantity of time to develop the experiment –two hours-. And the lack of experience working with WAFs and Codeigniter that all students had. We decided to avoid that students implemented some security aspects.

Figure 4.5 shows a class diagram that represents the classes involved in the application.

Figure 4-5: Experiment class diagram.



Iteration 1

The first iteration was simple; the main idea was to develop two views, the initial view displayed two options and the second view displayed information about the application creator.

We consider experimental groups (G2 and G3) should select and learn the next two concerns –it is important to highlight that the students had to select the concerns by themselves as they considered–.

- Display information on screen
- Routes and navigability

The following text was presented to the students:

> *The initial menu has two options (Manage Coffee Stores and Contact). Besides, it will display the software creator's information at 'Contact' section.*

Figures 4.6 and 4.7 show a solution from one subject to the iteration 1.

Figure 4-6: A solution from a subject to the initial menu of the iteration 1.

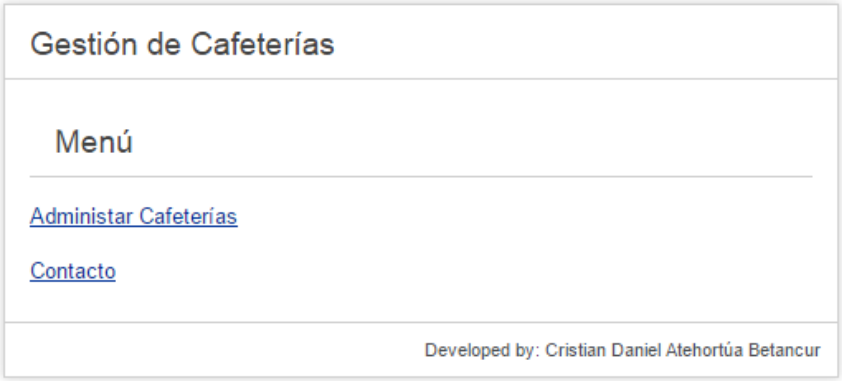
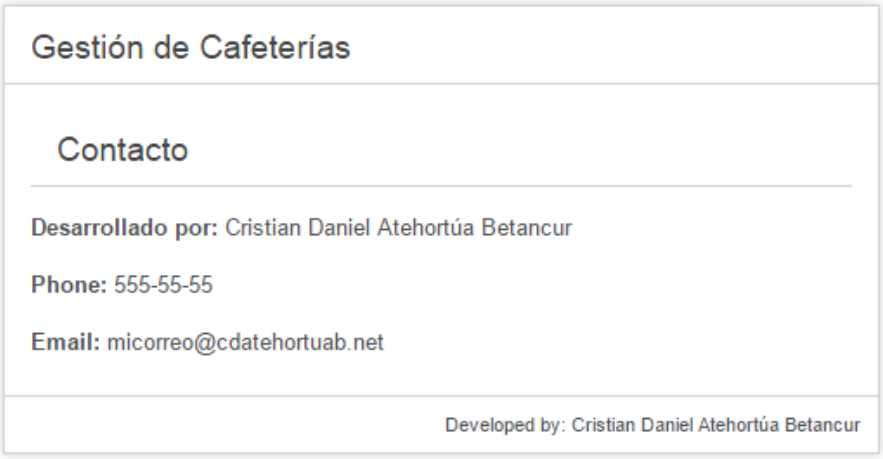


Figure 4-7: A solution from a subject to the contact section of the iteration 1.



Iteration 2

The second iteration was more difficult; the main idea was to understand how the mechanism to extract data from the database is and how to connect this data with the Codeigniter model layer. Finally, display this information in a view.

We consider experimental groups (G2 and G3) should select and learn the next four concerns.

- Display information on screen
- Routes and navigability
- Capture and data assignment
- Data Selection

The following text was presented to the students:

> *All coffee stores will be displayed at 'Manage Coffee Stores' section. It's required to display the id, name, physical address, city and state of all coffee stores.* ***Note:*** *there is a .sql file with the coffee stores information (see Appendix E).*

Figure 4.8 shows a solution from one subject to the iteration 2.

Figure 4-8: A solution from a subject to the iteration 2.

Gestión de Cafeterías

Cafeterías

ID	Nombre	Dirección Física	Ciudad	Estado
20152	Sweet Beach Cakery	2200 E 2nd St Bldg J	Gulf Shores	AL
20153	Heritage House Coffee & Tea	18 McFarland Blvd	Northport	AL
20154	Tasty Tea	Not found	Alabaster	AL
20155	Southern Decadence Desserts	1956B S University Blvd	Mobile	AL
20156	Florist Plus	7938 Vaughn Rd	Montgomery	AL
20157	Sonny's Catfish Cafe'	815 County Road 61	Houston	AL
20158	Chat A Way Cafe	4366 Old Shell Rd	Mobile	AL
20159	Starbucks Coffee	801 20th St S	Birmingham	AL
20160	O'Henry's Coffee	2831 18th St S	Birmingham	AL
20161	Red Diamond Coffee & Tea	Not found		ot
20162	Starbucks Coffee	1510 Government St	Mobile	AL
20163	Starbucks Coffee	Not found		ot
20164	G N U's Room	414 S Gay St	Auburn	AL
20165	Starbucks Coffee	1015 Memorial Pkwy NW	Huntsville	AL
20166	Sunset Cafe	203 E Main St	Samson	AL
20167	Cafe On Main	110 2nd Ave W	Oneonta	AL
20168	Green Acres Cafe South	8500 1st Ave N	Birmingham	AL
20169	The Daily Perk	913 N Daleville Ave	Daleville	AL
20170	Starbucks Coffee	2000 Riverchase Galleria	Birmingham	AL
20171	Starbucks Coffee	2056 Interstate Dr	Opelika	AL
20172	Seattle's Best Coffee	2601 Mamie L Foster	Birmingham	AL
20173	American Coffee House And Company	22229 Highway 31	Flomaton	AL
20174	Carpe Diem Coffee & Tea Company	4072 Old Shell Rd	Mobile	AL
20175	Bay Breeze Cafe	50 S Church St # D	Fairhope	AL
20176	Starbucks Coffee	3255 Airport Blvd	Mobile	AL
20177	Starbucks Coffee	1650 US Highway 98	Daphne	AL
20178	Bee Hive	11 W Claiborne St	Monroeville	AL
20179	Tomek's	2320 2nd Ave N	Birmingham	AL
20180	Chelsea Coffee House	109 Foothills Pkwy	Chelsea	AL
20181	Mokas Coffee House Inc	1204 Shelton Beach Rd Ste 1	Saraland	AL

Developed by: Cristian Daniel Atehortúa Betancur

Iteration 3

The main idea of this iteration was to understand how the mechanism to delete data from database works.

We consider experimental groups (G2 and G3) should select and learn the next four concerns.

- Display information on screen
- Routes and navigability
- Capture and data assignment
- Deleting Data

The following text was presented to the students:

> *In the previous list each coffee store will have a delete button. When is pressed, it will delete the coffee store in the database and will display a message "the coffee store was successfully deleted".*

Figures 4.9 and 4.10 show a solution from one subject to the iteration 3.

Figure 4-9: A solution from a subject to the delete coffee store message of iteration 3.

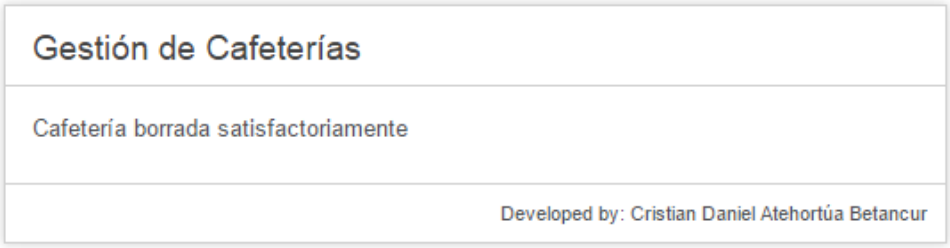


Figure 4-10: A solution from a subject to the delete button of iteration 3.

Gestión de Cafeterías

Cafeterías

ID	Nombre	Dirección Física	Ciudad	Estado	
20152	Sweet Beach Cakery	2200 E 2nd St Bldg J	Gulf Shores	AL	Borrar
20153	Heritage House Coffee & Tea	18 McFarland Blvd	Northport	AL	Borrar
20154	Tasty Tea	Not found	Alabaster	AL	Borrar
20155	Southern Decadence Desserts	1956B S University Blvd	Mobile	AL	Borrar
20156	Florist Plus	7938 Vaughn Rd	Montgomery	AL	Borrar
20157	Sonny's Catfish Cafe'	815 County Road 61	Houston	AL	Borrar
20158	Chat A Way Cafe	4366 Old Shell Rd	Mobile	AL	Borrar
20159	Starbucks Coffee	801 20th St S	Birmingham	AL	Borrar
20160	O'Henry's Coffee	2831 18th St S	Birmingham	AL	Borrar
20161	Red Diamond Coffee & Tea	Not found		ot	Borrar
20162	Starbucks Coffee	1510 Government St	Mobile	AL	Borrar
20163	Starbucks Coffee	Not found		ot	Borrar
20164	G N U's Room	414 S Gay St	Auburn	AL	Borrar
20165	Starbucks Coffee	1015 Memorial Pkwy NW	Huntsville	AL	Borrar
20166	Sunset Cafe	203 E Main St	Samson	AL	Borrar
20167	Cafe On Main	110 2nd Ave W	Oneonta	AL	Borrar
20168	Green Acres Cafe South	8500 1st Ave N	Birmingham	AL	Borrar
20169	The Daily Perk	913 N Daleville Ave	Daleville	AL	Borrar
20170	Starbucks Coffee	2000 Riverchase Galleria	Birmingham	AL	Borrar
20171	Starbucks Coffee	2056 Interstate Dr	Opelika	AL	Borrar
20172	Seattle's Best Coffee	2601 Mamie L Foster	Birmingham	AL	Borrar
20173	American Coffee House And Company	22229 Highway 31	Flomaton	AL	Borrar
20174	Carpe Diem Coffee & Tea Company	4072 Old Shell Rd	Mobile	AL	Borrar
20175	Bay Breeze Cafe	50 S Church St # D	Fairhope	AL	Borrar
20176	Starbucks Coffee	3255 Airport Blvd	Mobile	AL	Borrar
20177	Starbucks Coffee	1650 US Highway 98	Daphne	AL	Borrar
20178	Bee Hive	11 W Claiborne St	Monroeville	AL	Borrar
20179	Tomek's	2320 2nd Ave N	Birmingham	AL	Borrar
20180	Chelsea Coffee House	109 Foothills Pkwy	Chelsea	AL	Borrar
20181	Mokas Coffee House Inc	1204 Shelton Beach Rd Ste 1	Saraland	AL	Borrar

Developed by: Cristian Daniel Atehortúa Betancur

Iteration 4

In the last iteration, the students had to understand how to extract elements from a table by using the proper filter mechanism and create a proper view to display them.

We consider experimental groups (G2 and G3) should select and learn the next four concerns.

- Display information on screen
- Routes and navigability
- Capture and data assignment
- Data selection using filters

The following text was presented to the students:

> *Finally, in the previous list each coffee store name will have a link. When is pressed the name, it will be displayed: a list of the baristas associated to the coffee store selected (with their respective id, name, email and phone).* ***Note:*** *there is a .sql file with the baristas information (see Appendix E).*

Figures 4.11 and 4.12 show a solution from one subject to the iteration 4.

Figure 4-11: A solution from a subject to the coffee store name link of iteration 4.

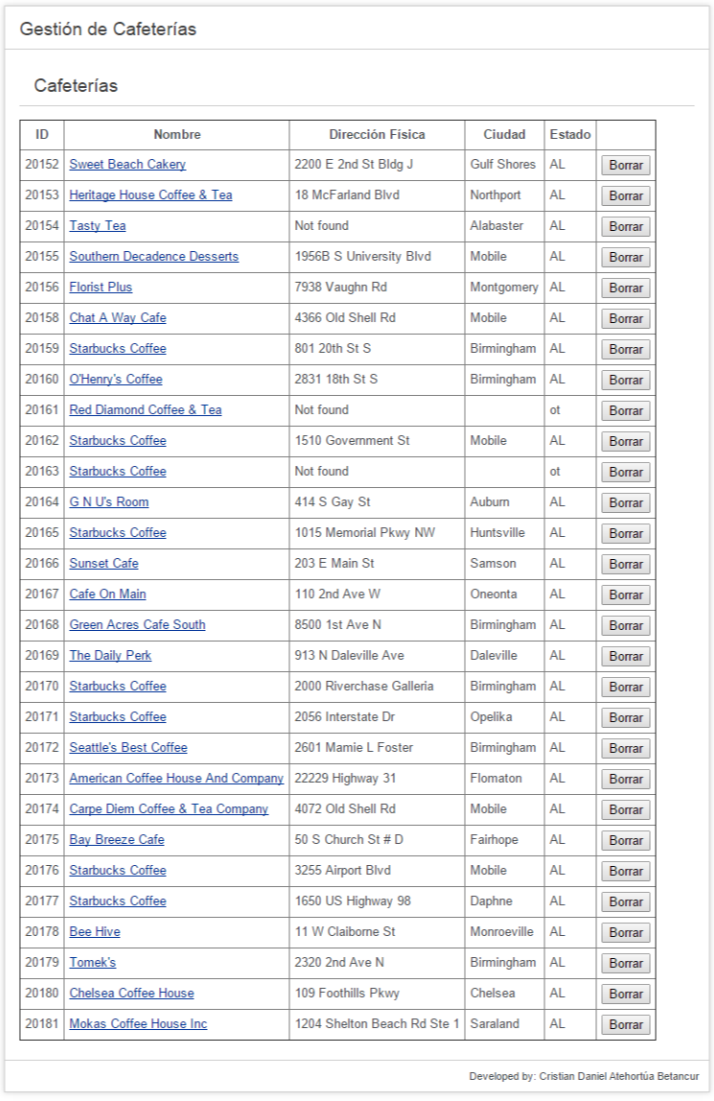
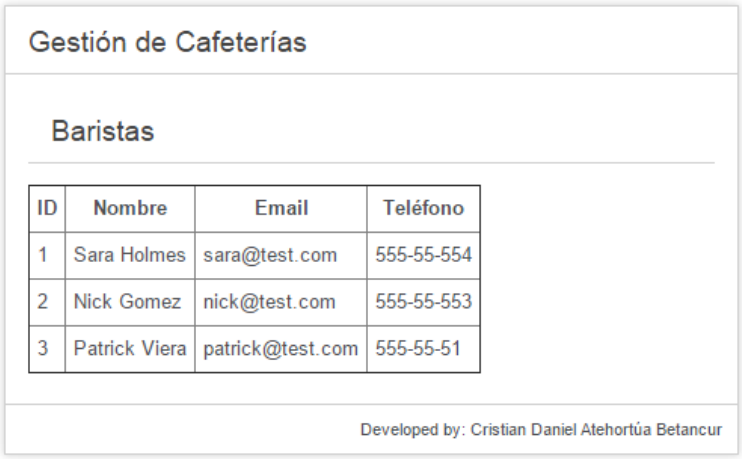


Figure 4-12: A solution from a subject to the barista list display of iteration 4.



4.2.8 Post-questionnaire

At the end of the experiment, the students were submitted to the post-questionnaire (see Appendix F). The answers from all the participants are detailed in Appendix G and further analyses later in this chapter.

4.3 Data analysis

As previously described, the subjects were submitted to a pre-questionnaire to screen out possible background and basic skills deviations. During the experiment, the task completion time was recorded by supervisors, and finally, a post-questionnaire collected further data. This section presents a detailed analysis of the collected data in order to provide evidence of the validity of the assumptions presented by this thesis. Firstly, it will be shown that all groups have no significant background deviations and the acquaintance of the Codeigniter framework is correctly assumed in some cases. Secondly, the analysis will focus on comparing results between the Baseline (G1) and Experimental Groups (G2 and G3).

4.3.1 Statistical relevance

To provide statistical relevance in the analysis of the questionnaires items, the results are interpreted as described next. Let the null hypothesis be denoted as H_0, the alternative hypothesis as H_1, the baseline group as G_b, the experimental group as G_e and p the

probability estimator of wrongly rejecting the null hypothesis. Then, the alternative hypothesis are either: (i) H_1:$G_e \neq G_b$, the experimental group differs from the baseline, (ii) H_1:$G_e < G_b$, the measure in the experimental group is lower than the baseline, or (iii) H_1:$G_e > G_b$, the measure in the experimental group is greater than the baseline. The outcomes of the two treatments were compared for every answer using the non-parametric, two-sample, rank-sum Wilcoxon-Mann-Whitney [64] test, with $n_1=5$ and $n_2=5$ –both experimental groups (G2 and G3) have 5 subjects–.

The significance level for all tests was set to 5%, so probability values of $p \leq 0.05$ are considered significant, and $p \leq 0.01$ considered highly significant. The corresponding alternative hypothesis are further detailed for each question, and a summary of the base statistics and corresponding test values can be found in Appendices C and G.

4.3.2 Background

Although an objective comparison between the background of each group was already performed using the subjects average grades in key courses (section 4.2.2), this section rejects any subjective difference amongst the participants with respect to their basic skills.

Table 4-4: Summary of Background results between the Baseline (G1) and Experimental Group 2 (G2), including the values of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 2		Group 1			
	$\bar{x}$	σ	$\bar{x}$	σ	H1	p-value
Question 1	1,00	0,00	1,00	0,00	≠	1,000
Question 2	1,80	0,45	1,40	0,55	≠	0,524
Question 3	2,40	0,55	2,20	1,30	≠	0,683
Question 4	3,00	0,00	2,80	0,84	≠	0,444
Question 5	3,40	0,55	2,80	1,10	≠	0,643
Question 6	3,60	0,89	3,80	0,84	≠	0,921
Question 7	3,00	1,00	3,20	1,10	≠	0,810
Question 8	3,60	0,55	4,00	0,71	≠	0,643
Question 9	2,40	1,14	3,00	1,41	≠	0,460
Question 10	3,00	0,71	2,80	0,84	≠	0,881

Table 4-5: Summary of Background results between the Baseline (G1) and Experimental Group 3 (G3), including the values of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 3		Group 1			
	$\bar{x}$	σ	$\bar{x}$	σ	H1	p-value
Question 1	1,00	0,00	1,00	0,00	≠	1,000
Question 2	1,20	0,45	1,40	0,55	≠	1,000
Question 3	2,20	1,30	2,20	1,30	≠	1,000
Question 4	2,80	0,45	2,80	0,84	≠	1,000
Question 5	3,80	0,45	2,80	1,10	≠	0,167
Question 6	4,00	0,71	3,80	0,84	≠	0,881
Question 7	2,60	0,89	3,20	1,10	≠	0,365
Question 8	4,00	0,71	4,00	0,71	≠	1,000
Question 9	3,00	1,22	3,00	1,41	≠	1,000
Question 10	3,00	0,00	2,80	0,84	≠	0,444

BG1 I have considerable experience developing in Codeigniter

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 1.000) in the scores for the experimental group G2 ($\bar{x}$ = 1.00, σ = 0.00) and baseline G1 ($\bar{x}$ = 1.00, σ = 0.00) conditions, as seen in Table 4.4. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 1.000) in the scores for the experimental group G3 ($\bar{x}$ = 1.00, σ = 0.00) and baseline G1 ($\bar{x}$ = 1.00, σ = 0.00) conditions, as seen in Table 4.5. As expected, the students didn't have acquaintance working with Codeigniter. This was a mandatory condition to the effective prosecution of the experiment goals.

BG2 I have considerable experience using frameworks

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.524) in the scores for the experimental group G2 ($\bar{x}$ = 1.80, σ = 0.45) and baseline G1 ($\bar{x}$ = 1.40, σ = 0.55) conditions, as seen in Table 4.4. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 1.000) in the scores for the experimental group G3 ($\bar{x}$ = 1.20, σ = 0.45) and baseline G1 ($\bar{x}$ = 1.40, σ = 0.55) conditions, as seen in Table 4.5. As expected, the students didn't have acquaintance not only with Codeigniter but also working with frameworks. This was a big challenge for novice WAF developers, due to their complete lack of knowledge with the use of these tools.

BG3 I have considerable experience developing web application

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.683) in the scores for the experimental group G2 ($\bar{x}$ = 2.40, σ = 0.55) and baseline G1 ($\bar{x}$ = 2.20, σ = 1.30) conditions, as seen in Table 4.4. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 1.000) in the scores for the experimental group G3 ($\bar{x}$ = 2.20, σ = 1.30) and baseline G1 ($\bar{x}$ = 2.20, σ = 1.30) conditions, as seen in Table 4.5. Similar to BG1.2, this item shows a low average. This result also serves to show the lack of experience developing web application. Most students haven't worked in companies which makes them in useful novice developers to develop the experiment.

BG4 I have considerable experience programming in PHP

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.444) in the scores for the experimental group G2 ($\bar{x}$ = 3.00, σ = 0.00) and baseline G1 ($\bar{x}$ = 2.80, σ = 0.84) conditions, as seen in Table 4.4. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 1.000) in the scores for the experimental group G3 ($\bar{x}$ = 2.80, σ = 0.45) and baseline G1 ($\bar{x}$ = 2.80, σ = 0.84) conditions, as seen in Table 4.5. This item discarded the PHP language as a validation threat to the reliability of the experiment results. Although the students had to work with PHP, the main idea was the students had to use the Codeigniter components to develop the experiment tasks, which were detailed at the different learning materials.

BG5 I have considerable experience using MySQL

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.643) in the scores for the experimental group G2 ($\bar{x}$ = 3.40, σ = 0.55) and baseline G1 ($\bar{x}$ = 2.80, σ = 1.10) conditions, as seen in Table 4.4. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 0.167) in the scores for the experimental group G3 ($\bar{x}$ = 3.80, σ = 0.45) and baseline G1 ($\bar{x}$ = 2.80, σ = 1.10) conditions, as seen in Table 4.5. To solve some tasks, the students had to use SQL language or the proper Codeigniter functions. This item discarded this as a possible threat to the validity of results.

BG6 I have considerable experience with object-oriented programming

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.921) in the scores for the experimental group G2 ($\bar{x}$ = 3.60, σ = 0.89) and baseline G1 ($\bar{x}$ = 3.80, σ = 0.84)

conditions, as seen in Table 4.4. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 0.881) in the scores for the experimental group G3 ($\bar{x}$ = 4.00, σ = 0.71) and baseline G1 ($\bar{x}$ = 3.80, σ = 0.84) conditions, as seen in Table 4.5. Consistent with their academic track, all groups exhibited a positive response. Object-oriented programming was a crucial aspect in order to develop the experiment.

BG7 I have considerable experience with agile development methodologies

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.810) in the scores for the experimental group G2 ($\bar{x}$ = 3.00, σ = 1.00) and baseline G1 ($\bar{x}$ = 3.20, σ = 1.10) conditions, as seen in Table 4.4. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 0.365) in the scores for the experimental group G3 ($\bar{x}$ = 2.60, σ = 0.89) and baseline G1 ($\bar{x}$ = 3.20, σ = 1.10) conditions, as seen in Table 4.5. This item served as an evaluation of the students' feelings towards the iterative development that would characterize the experiment process. As such, the development methodology proved not to be an obstacle throughout the experiment.

BG8 I have considerable experience with UML diagrams

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.643) in the scores for the experimental group G2 ($\bar{x}$ = 3.60, σ = 0.55) and baseline G1 ($\bar{x}$ = 4.00, σ = 0.71) conditions, as seen in Table 4.4. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 1.000) in the scores for the experimental group G3 ($\bar{x}$ = 4.00, σ = 0.71) and baseline G1 ($\bar{x}$ = 4.00, σ = 0.71) conditions, as seen in Table 4.5. Consistent with their academic track, all groups exhibited a positive response. Even when UML diagrams were not presented to the students, this element could be useful for some students in order to better-understand the tables and classes they had to use.

BG9 I have considerable experience analyzing and specifying information systems

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.460) in the scores for the experimental group G2 ($\bar{x}$ = 2.40, σ = 1.14) and baseline G1 ($\bar{x}$ = 3.00, σ = 1.41) conditions, as seen in Table 4.4. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 1.000) in the scores for the experimental group G3 ($\bar{x}$ = 3.00, σ = 1.22) and baseline G1 ($\bar{x}$ = 3.00, σ = 1.41) conditions, as seen in Table 4.5. Almost since the beginning of their

academic track, the students engage in the analysis and specification of information systems. The medium-low average results could be explained due to the lack of experience working in real companies that most of the students presented. This item also serves to remark the role of 'novice developers' that students presented in the experiment.

BG10 I have considerable experience analyzing and implementing databases

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.881) in the scores for the experimental group G2 ($\bar{x}$ = 3.00, σ = 0.71) and baseline G1 ($\bar{x}$ = 2.80, σ = 0.84) conditions, as seen in Table 4.4. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 0.444) in the scores for the experimental group G3 ($\bar{x}$ = 3.00, σ = 0.00) and baseline G1 ($\bar{x}$ = 2.80, σ = 0.84) conditions, as seen in Table 4.5. Similar to BG1.5, to solve some tasks, the students had to implement a database and access it and modify it. The tasks that involved this aspect were too simple which means a low acquaintance with working with databases was enough. This item discarded this as a possible threat to the validity of results.

4.3.3 External Factors

The experiment environment was an important concern. In a common working place there are aspects out of control (inter-participants interaction, disturbances, noise, etc.), so the main idea was to discard possible validation threatening environmental factors.

Table 4-6: Summary of external factors results between the Baseline (G1) and Experimental Group 2 (G2), including the values of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 2		**Group 1**			
	$\bar{x}$	**σ**	$\bar{x}$	**σ**	**H1**	**p-value**
External Factor 1	2,40	1,52	2,00	1,22	≠	0,643
External Factor 2	5,00	0,00	4,00	1,00	≠	0,167
External Factor 3	1,60	0,89	2,20	1,10	≠	0,524

Table 4-7: Summary of external factors results between the Baseline (G1) and Experimental Group 3 (G3), including the values of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 3		Group 1			
	$\bar{x}$	σ	$\bar{x}$	σ	H1	p-value
External Factor 1	2,00	1,73	2,00	1,22	≠	0,921
External Factor 1	4,80	0,45	4,00	1,00	≠	0,286
External Factor 3	1,20	0,45	2,20	1,10	≠	0,167

EF1 I found the whole experience environment intimidating

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.643) in the scores for the experimental group G2 ($\bar{x}$ = 2.40, σ = 1.52) and baseline G1 ($\bar{x}$ = 2.00, σ = 1.22) conditions, as seen in Table 4.6. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 0.921) in the scores for the experimental group G3 ($\bar{x}$ = 2.00, σ = 1.73) and baseline G1 ($\bar{x}$ = 2.00, σ = 1.22) conditions, as seen in Table 4.7. Overall, the participants didn't found the experience intimidating. This item served to discard this factor, as shown by the low scores exhibited by all participants.

EF2 I enjoyed programming and trying to develop the application during the experiment

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.167) in the scores for the experimental group G2 ($\bar{x}$ = 5.00, σ = 0.00) and baseline G1 ($\bar{x}$ = 4.00, σ = 1.00) conditions, as seen in Table 4.6. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 0.286) in the scores for the experimental group G3 ($\bar{x}$ = 4.80, σ = 0.45) and baseline G1 ($\bar{x}$ = 4.00, σ = 1.00) conditions, as seen in Table 4.7. This item measured the fun factor. There was a common sense of a positive feeling towards the experiment so this factor can be discarded as a threat to the whole experiment.

EF3 I felt distracted by other students during the experiment

Let H_1:$G_{e2} \neq G_b$, there was no significant difference (p = 0.524) in the scores for the experimental group G2 ($\bar{x}$ = 1.60, σ = 0.89) and baseline G1 ($\bar{x}$ = 2.20, σ = 1.10) conditions, as seen in Table 4.6. Let H_1:$G_{e3} \neq G_b$, there was no significant difference (p = 0.167) in the scores for the experimental group G3 ($\bar{x}$ = 1.20, σ = 0.45) and baseline G1 ($\bar{x}$ = 2.20, σ = 1.10) conditions, as seen in Table 4.7. In a familiar, non-intimidating setting,

it is easier to interact and to vocalize more, producing more noise and increasing the disturbance level. This item served to discard this factor, as shown by the low scores exhibited by all participants.

4.3.4 Overall satisfaction

This group of questions was intended to provide subjective validation to the thesis on an overall scope, by questioning subjects on their performance, comfort and feel for the presented learning environment and material.

Table 4-8: Summary of overall satisfaction results between the Baseline (G1) and Experimental Group 2 (G2), including the values of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 2		Group 1			
	$\bar{x}$	σ	$\bar{x}$	σ	H1	p-value
Overall Satisfaction 1	3,80	0,84	2,60	0,89	>	0,028
Overall Satisfaction 2	4,00	0,71	2,40	0,55	>	0,012
Overall Satisfaction 3	2,60	0,89	4,60	0,55	<	0,004
Overall Satisfaction 4	2,60	1,52	4,40	0,55	<	0,036

Table 4-9: Summary of overall satisfaction results between the Baseline (G1) and Experimental Group 3 (G3), including the values of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 3		Group 1			
	$\bar{x}$	σ	$\bar{x}$	σ	H1	p-value
Overall Satisfaction 1	4,20	0,84	2,60	0,89	>	0,028
Overall Satisfaction 2	4,20	1,10	2,40	0,55	>	0,024
Overall Satisfaction 3	2,20	1,10	4,60	0,55	<	0,004
Overall Satisfaction 4	2,00	1,00	4,40	0,55	<	0,004

OS1 I consider the time available to develop the experiment was adequate

Let H_1:G_{e2}>G_b, there was a significant difference (p = 0.028) in the scores for the experimental group G2 ($\bar{x}$ = 3.80, σ = 0.84) and baseline G1 ($\bar{x}$ = 2.60, σ = 0.89) conditions, as seen in Table 4.8. Let H_1:G_{e3}>G_b, there was a significant difference (p = 0.028) in the scores for the experimental group G3 ($\bar{x}$ = 4.20, σ = 0.84) and baseline G1 ($\bar{x}$ = 2.60, σ = 0.89) conditions, as seen in Table 4.9. This item could be analyzed as a relation with the amount of tasks of each group finished. As experimental groups G2 and

G3 were able to complete more tasks than Baseline G1 (see section 4.3.7), this aspect is traduced in more satisfaction with the time available to develop the experiment. Furthermore, G1 participants were able to complete fewer tasks which traduces in less satisfaction with the time available to develop the experiment.

OS2 I consider the documentation available to be sufficient to develop the experiment

Let H_1:G_{e2}>G_b, there was a significant difference (p = 0.012) in the scores for the experimental group G2 ($\bar{x}$ = 4.00, σ = 0.71) and baseline G1 ($\bar{x}$ = 2.40, σ = 0.55) conditions, as seen in Table 4.8. Let H_1:G_{e3}>G_b, there was a significant difference (p = 0.024) in the scores for the experimental group G3 ($\bar{x}$ = 4.20, σ = 1.10) and baseline G1 ($\bar{x}$ = 2.40, σ = 0.55) conditions, as seen in Table 4.9. A typical issue of software development in general is that there is never enough and/or good documentation. However, the intent of this item was to perceive if the new WAF learning technique and its DL application would improve the usage and value of the available documentation, deemed sufficient to effectively undertake all the tasks presented. The exhibited scores give a strong support of that assumption. Moreover, the score regarding Baseline G1 vs. G2 and G1 vs G3 is pretty clear to show that the new WAF learning technique helped dealing with WAF learning. G3 presented a higher average ($\bar{x}$ = 4.20) satisfaction with the documentation than G2 ($\bar{x}$ = 4.00); and it could be explained because G3 had the complete WAF learning material and G2 only the meso-tasks or examples.

OS3 I felt the need to have access to more information on how to use the framework

Let H_1:G_{e2}<G_b, there was a highly significant difference (p = 0.004) in the scores for the experimental group G2 ($\bar{x}$ = 2.60, σ = 0.89) and baseline G1 ($\bar{x}$ = 4.60, σ = 0.55) conditions, as seen in Table 4.8. Let H_1:G_{e3}<G_b, there was a highly significant difference (p = 0.004) in the scores for the experimental group G3 ($\bar{x}$ = 2.20, σ = 1.10) and baseline G1 ($\bar{x}$ = 4.60, σ = 0.55) conditions, as seen in Table 4.9. Even when Codeigniter official documentation was plenty of pages and full of different descriptions of each element, the baseline G1 showed a strong need to access to more information. On the other hand, experimental groups showed less need to access to more information. The scores obtained by this item give strong evidence that the presented WAF learning technique

proved to be well suited for more easily learning about a WAF; identifying from the beginning the specific material related with the subjects' requirements and giving useful examples to develop the different tasks.

OS4 Despite of my experience, the tools and documentation available, delayed my work considerably

Let H_1:G_{e2}<G_b, there was a significant difference (p = 0.036) in the scores for the experimental group G2 ($\bar{x}$ = 2.60, σ = 1.52) and baseline G1 ($\bar{x}$ = 4.40, σ = 0.55) conditions, as seen in Table 4.8. Let H_1:G_{e3}<G_b, there was a highly significant difference (p = 0.004) in the scores for the experimental group G3 ($\bar{x}$ = 2.00, σ = 1.00) and baseline G1 ($\bar{x}$ = 4.40, σ = 0.55) conditions, as seen in Table 4.9. Similar to OS3 subjects from baseline G1 felt the documentation was not the proper to complete the different tasks –even when the official documentation is the commonly used by most developers–. This could be compared with OS1 results, as time was running out and the tasks weren't completed the subjects felt the tools and the material was not the proper to the experiment. Similar to OS3, the scores obtained by this item give strong evidence that the new WAF learning technique proved to be well suited for more easily learning about a WAF.

4.3.5 Development process

This category of items intended to ascertain how hard it was to complete each of the tasks presented and its evolution throughout the experiment by using the different learning materials. Due to some subjects from some groups weren't able to complete some tasks; we decided to put the highest value –five– in those cases.

Table 4-10: Summary of development process results between the Baseline (G1) and Experimental Group 2 (G2), including the values of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 2		Group 1			
	$\bar{x}$	σ	$\bar{x}$	σ	H1	p-value
Development Process 1	1,80	1,30	3,20	1,48	<	0,103
Development Process 2	1,60	0,89	4,40	0,89	<	0,008
Development Process 3	2,00	1,00	5,00	0,00	<	0,004
Development Process 4	3,60	1,95	5,00	0,00	<	0,222

Table 4-11: Summary of development process results between the Baseline (G1) and Experimental Group 3 (G3), including the values of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 3		Group 1			
	$\bar{x}$	σ	$\bar{x}$	σ	H1	p-value
Development Process 1	1,20	0,45	3,20	1,48	<	0,024
Development Process 2	3,00	1,87	4,40	0,89	<	0,143
Development Process 3	3,60	1,95	5,00	0,00	<	0,222
Development Process 4	4,80	0,45	5,00	0,00	<	0,500

As expected, subjects from experimental groups presented lower scores than subjects from baseline group to the question “It was hard to find out how to use the framework to complete... iteration 1, 2, 3, 4” in all iterations. This assumption is strongly confirmed in a later analysis at section 4.3.7.

In an overall analysis, it can be stated:

- In the case of Baseline (G1) and experimental group 2 (G2) with H_1:G_{e2}<G_b, at development process 2 (iteration 2) there was a highly significant difference (p = 0.008) in the scores for the experimental group G2 ($\bar{x}$ = 1.60, σ = 0.89) and baseline G1 ($\bar{x}$ = 4.40, σ = 0.89) conditions, as seen in Table 4.10.
- In the case of Baseline (G1) and experimental group 2 (G2) with H_1:G_{e2}<G_b, at development process 3 (iteration 3) there was a highly significant difference (p = 0.004) in the scores for the experimental group G2 ($\bar{x}$ = 2.00, σ = 1.00) and baseline G1 ($\bar{x}$ = 5.00, σ = 0.00) conditions, as seen in Table 4.10.
- In the case of Baseline (G1) and experimental group 3 (G3) with H_1:G_{e3}<G_b, at development process 1 (iteration 1) there was a significant difference (p = 0.024) in the scores for the experimental group G3 ($\bar{x}$ = 1.20, σ = 0.45) and baseline G1 ($\bar{x}$ = 3.20, σ = 1.48) conditions, as seen in Table 4.11.

The scores obtained by this item give strong evidence that the new WAF learning technique proved to be well suited for more easily learning about a WAF.

4.3.6 Framework knowledge

In order to measure the increase in framework knowledge, a set of 10 questions were presented to the subjects at the end of the experiment. These questions intended to ascertain how much correct information about the WAF the participants had acquired. The questions were related with the concepts to the different components involved to the design of the macro-task at section 4.2.7. It was assumed that all groups had no prior knowledge of the WAF, as corroborated by item BG1.

Table 4-12: Framework knowledge questions and answers. All items were presented as true-false statements.

Question	1	2	3	4	5	6	7	8	9	10
Answer	F	F	T	T	F	T	T	F	T	F

Results

The relevance of an item-to-item analysis of the scores isn't so much important as the total amount of knowledge the subjects acquired. So, the results are shown aggregated and processed as the total knowledge acquired.

Table 4-13: Framework knowledge group statistics.

Group	N	Mean	Standard Deviation	Standard Error Mean
G1 - Baseline	5	3,000	1,871	0,8367
G2	5	4,400	1,140	0,5099
G3	5	5,000	1,581	0,7071

Table 4.13 shows subjects from baseline group 1 had an average of 3 correct answers of 10 questions; G2 had an average of 4.4/10 and G3 5/10. The results provide evidence that the new WAF learning technique support the hypothesis that it helps novices on learning about a WAF.

Similar to OS2 results, G3 showed more WAF knowledge than G2. It could be explained because G3 had the complete WAF learning material and G2 only the meso-tasks or examples.

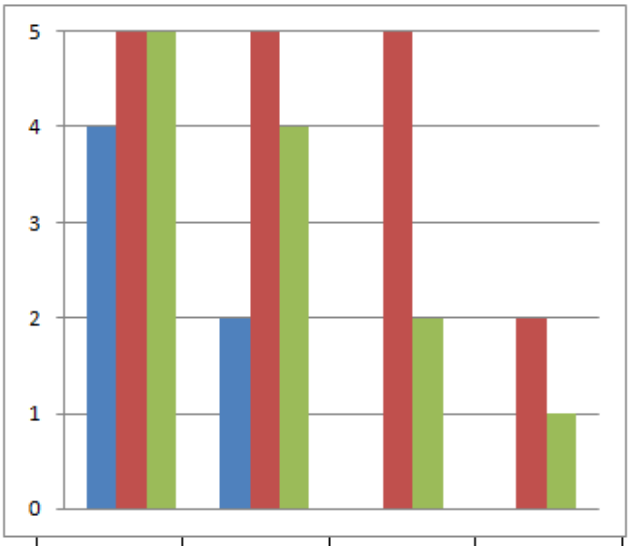
4.3.7 Objective measurement

During the experiment, the duration each subject took to complete each iteration was recorded. At the end, these results were processed (see the complete results at Appendix H). However, the quantity of iterations completion by baseline group was very low –only six from twenty– and it made difficult to analyze the time between the groups. Due to iteration 1 was the most completed by subjects from baseline group, we decided to make a time analysis focusing only in iteration 1. Table 4.14 shows the iteration 1 completion time results (average per group). This result shows a significant time reduction of approximately 40% between the experimental groups' iteration 1 completion and the baseline group.

Table 4-14: Iteration 1 completion time results (average per group). Units in minutes.

	Iteration 1
Baseline - G1	76
Experimental Group 2	30,2
Experimental Group 3	31,2

Figure 4-13: Number of subjects of each group who completed each iteration.



	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Baseline - G1	4	2	0	0
Experimental Group 2	5	5	5	2
Experimental Group 3	5	4	2	1

Table 4-15: Number of iterations completion (average per group).

Group	Iterations	Mean
G1 - Baseline	4	1,200
G2	4	3,400
G3	4	2,400

Figure 4.13 and table 4.15 highlight a very important aspect. As expected the average of iterations completion of the experimental groups G2 ($\bar{x}$ = 3.40) and G3 ($\bar{x}$ = 2.40) were higher than the baseline G1 ($\bar{x}$ = 1.20). This result gives strong evidence that the presented new WAF learning technique helps developers not only with their first contact with the framework but also in all the development process.

4.4 Validation threats

The outcome of validation is to gather enough scientific evidence to provide a sound interpretation of the results. Validation threats are issues and scenarios that may distort that evidence and thus incorrectly support (or discard) expected results. Each validation threat should be expected and addressed a priori in order to yield unbiased results or, at least, minimized a posteriori with effective counter-measures.

This section addresses expected validation threats and how these were discarded, while others should be attentively focused in future experiments.

- **Insufficient skills to execute the tasks.** The tasks required participants to have the necessary skill to build and evolve information systems, namely knowing how to work with the given programming language, IDE and database engine. Once again, this threat was discarded by both pre-experiment evaluation and pre-experiment questionnaire, through items BG4, BG5, BG6, BG8, BG9 and BG10.

- **Experiment-related factors.** Knowingly being part of an experiment changes the mood and may be an inhibitor of normal development. The performance may be conditioned by the feel of being observed and judged. The results of item EF1 allowed this threat to be discarded.

- **Environment factors.** Despite the noise or disturbance that could be generated in a laboratory by other students, it was necessary to make sure that the experiment environment wasn't a threat to validity. Item EF3 discarded this threat.

- **Lack of motivation.** Due to the length of the tasks (experiment went about 2 hours), and the fact that there was no compensation to individuals participating in the experiment, the lack of motivation could hinder the outcome. This threat is discarded by item EF2.

The following threats were not completely discarded, and should be the focus of future studies:

- **Similarity between the meso-tasks and the macro-task iterations:** even when was developed a study case establishing the common concerns developers have developing web applications (see section 3.2), is difficult to prove if experimental groups were able to develop more iterations than baseline group because of the similarity between the experiment tasks and the meso-tasks documentation. But the important point is that this type of learning documentation could serve as a base to develop a wide range of applications.

- **A quasi-experiment and the new WAF learning technique applied only over a specific WAF:** even when was developed a study case establishing the generic WAFs components and micro-tasks over six different MVC WAFs (see section 3.1), is difficult to prove that these components and micro-tasks are the same to the whole range of available WAFs. Some studies especially in Java WAFs must be developed. Besides, more experiments over more WAFs must be applied.

The power of this study could also be improved by (i) increasing the number of participants, (ii) switching the participants roles, where individuals in the experimental groups would undergo the baseline process and vice-versa, (iii) developing more difficult tasks, (iv) integrating WAF experts subjects to the experiment, and (v) developing tasks not very similar to meso-tasks.

4.5 Summary

This chapter detailed a laboratory case –or quasi-experiment– conducted within a controlled experimental environment using the new WAF learning technique presented in previous chapters and the WAF official documentation over Codeigniter. There were three groups –Baseline (G1), Experimental Group 2 (G2) and Experimental Group 3 (G3)– to which their background and basic skills were screened through a pre-experiment questionnaire, guaranteeing no statistical deviation. All three groups went through the development of the same four incrementally tasks by using an unknown WAF (Codeigniter), each group used a different learning technique to enable a comparison between these techniques. A post-questionnaire and their number of iterations completion were used to assess the outcome of the experiment.

The final results support the hypothesis that the new WAF learning technique helps novices to more effectively learn about a WAF. Both experimental groups fared better at: number of iterations completion, knowledge intake and time, when compared to the Baseline group.

Some threats to this validation were identified and later discarded by analyzing the results in pre- and post-experiment questionnaires and due to the nature of the experimental setting.

5. Conclusions

Developing web systems is a complex, time-consuming, and expensive task that often requires the coordination of efforts across organizational and technical boundaries. Web Applications Frameworks (WAFs) provide different elements and components to develop effective web systems. They are powerful techniques for large-scale reuse promoting developers to improve quality and save costs and time.

Developers usually face the need for developing an application by using a specific WAF (perhaps an unknown one); consequently, they need to learn how to use the WAF for developing the application. Currently, when a developer has to use a specific WAF, he/she has to invest considerable effort and time on understanding it. This problem is due to the big amount of WAF components and the increasing number of documents. Sometimes, developers face the reading of hundreds of documentation pages with information they're never going to use. The main objective of the developers is to build web applications which have different requirements from one to another. By related the documentation to the developer concerns, we reduce the amount of documents they have to face, and they focus on what they need. We also improve this learning technique with the use of examples, providing a source of code reuse and a base for developing a wide range of applications.

So, to deal with these problems we combine: separation of concerns, micro-learning and example-based learning. Finally, we developed: (i) a list of web application concerns, (ii) a list of WAF components, (iii) a connection between concerns and WAF components, (iv) a list of meso-tasks or example for each concern, and (v) a web application to drive the WAF learning. With all of this, we designed a new WAF learning technique which improves the WAF learning, reduces the time and helps novice developers to drive their own WAF learning.

Evidence was collected that verifies the benefits behind these contributions and helps on the validation of the presented work.

5.1 Key Contributions

Briefly, the main contributions of the work presented in this thesis are:

- **Definition of a list of WAF components.** WAFs have some similarities and share some components. Based on this fact and previous studies, we proposed a list of 13 main components. We defined some tasks that a developer should learn to develop for each component. This list serves to better-understand how WAFs works, it describes from a high level to a very low level the main WAF elements. Some authors could use these components for some WAF comparison studies.

- **Definition of a list of developers' common web application concerns.** In the software development context, a concern is a particular goal, concept, or area of interest. Based on this perspective, we have faced the driving WAF learning by using a separation of concerns. In this approach, each concern represents an application feature supporting a kind of application requirements. Previous analysis shows that no matter how different seem some application from one another, they use similar concerns. We defined 29 concerns and we categorize them in different groups. When developers focus on their own concerns they save time and find what really they need to learn. Some authors could use these concerns for some comparison studies.

- **Definition of a list of meso-tasks or example for web application concerns.** Programmers frequently use a copy-and-paste process to develop their applications. We create a list of examples for each web application concern. When these examples are developed by WAF senior developers, quality and security are improved. Besides examples could serve as a base of WAF learning gluing together some WAF components and micro-tasks. Finally, this example list has been very detailed which means some authors could perform a comparison study between different WAFs.

- **Definition of a new WAF learning technique.** By join separation of concerns, micro-learning and example based learning we developed a new WAF learning technique. This technique is divided in several stages: (i) The developer has to extract his/her

application requirement, (ii) the developer has to select over a web application the concerns related with his/her requirements, (iii) the web application displays the micro-tasks documentation related with developer's requirements, (iv) the web application displays the meso-tasks or examples related with developer's requirements, and (v) the developer use that material in order to develop the web application.

- **Tool support for the WAF learning process.** DL application supports the previous contribution. This application allows displaying WAF learning material and allows senior WAF developers to insert, modify and delete the different learning material.

- **Impact study of the key benefits of the best practices and learning process through a laboratory case.** The impacts that the new WAF learning technique have on learners and developers, was ascertained through a controlled laboratory case or (quasi-) experiment. Evidence was collected that verifies the benefits behind these contributions.

5.2 Future work

The following are consider important research paths by the author.

- **Improve DL application.** DL application could also be improved allowing forum discussions and star rating documentation, also increasing the amount of material. Another interesting idea is to integrate this application with a wiki, allowing an easier way to modify and keep updated the documentation.

- **Develop more experiment studies.** The quasi-experiment developed was focus on novice WAF developers, but how this technique improves the learning of expert WAF developers is an important question. Developing more studies in professional –no academic- environments with developers engaged in full-scale software projects with defined time frames and development process will give powerful results.

- **Implement the learning technique in other areas.** Could this learning technique be implemented in other scenarios? Software tools in general? Implementing this

technique in other areas could bring new insights and could improve the learning experience.

A. Appendix: Pre-experiment Subject Data

Table A-1: Student grades for all participating groups. Each column represents the following courses: (I) Programming Fundamentals, (II) Data Structures, (III) Programming Object Oriented, (IV) Software Engineering, (V) Databases I, (VI) Programming Logical and Functional, and (VII) Requirements Engineering.

	I	II	III	IV	V	VI	VII	MEAN	DESV
G1 Subject 1	3,8	3,8	3,7	4,1	3,2	3,7	4,4	3,81	0,37
G1 Subject 2	4,5	3	3	4,6	*	3,8	4,1	3,83	0,71
G1 Subject 3	4,7	4	3,6	4,5	3,2	4,9	4,1	4,14	0,61
G1 Subject 4	5	4,8	3,4	4,2	3,7	4,8	3,8	4,24	0,63
G1 Subject 5	5	4,9	3,5	3,6	3,5	4,6	3,6	4,10	0,70
G2 Subject 1	4,2	3,5	*	3,8	3	3,5	*	3,60	0,44
G2 Subject 2	4,4	3,7	3,1	4,2	3,5	4,1	4,4	3,91	0,49
G2 Subject 3	5	4,9	4,7	4,3	4	4,7	4,4	4,57	0,35
G2 Subject 4	4,3	3,9	3,3	4,2	3,4	3,5	3,4	3,71	0,41
G2 Subject 5	3,8	3,5	3,2	3,7	3,2	3,3	3	3,39	0,29
G3 Subject 1	4	3,4	3,5	3,3	3,3	3,4	3,9	3,54	0,29
G3 Subject 2	4,5	3,8	4,1	3,7	3,4	3,5	3,2	3,74	0,44
G3 Subject 3	4,7	4,4	3,7	4,5	3,3	4	3,8	4,06	0,50
G3 Subject 4	4,2	3,7	3,6	4,1	3,6	4,8	3,8	3,97	0,43
G3 Subject 5	5	3,7	3	4,2	3	3,5	4	3,77	0,71

B. Appendix: Pre-questionnaire

The following is a copy of the anonymous questionnaire handed to the subjects of Group 1 Baseline (G1), Experimental Groups 2 and 3 (G2 and G3) before the beginning the experiment.

Estudio de técnicas de aprendizajes de frameworks de aplicación web

Agosto 2014

Pre-cuestionario:

Antes de iniciar el experimento, te invitamos a responder algunas preguntas sobre conocimientos generales en programación y desarrollo de software. Estas preguntas nos ayudarán a identificar tu perfil y tus conocimientos previos.

Para cada pregunta existen 5 opciones:

1. fuertemente en desacuerdo
2. en desacuerdo
3. no estoy ni de acuerdo ni en desacuerdo
4. estoy de acuerdo
5. estoy fuertemente de acuerdo

Identificador del Grupo: _____

Preguntas:

Tengo considerable experiencia con:

	1	2	3	4	5
... desarrollo en Codeigniter					
... el uso de frameworks de aplicación web					
... desarrollo de aplicaciones web					
... programación en PHP					
... programación en MySQL					
... programación orientada a objetos					
... el uso de metodologías agiles					
... el uso de diagramas UML					
... análisis y especificación de sistemas informáticos					
... análisis e implementación de base de datos					

C. Appendix: Pre-questionnaire answers

Table C-1: Pre-experiment questionnaire A results for Group 1 Baseline (G1) and Experimental Group 2 (G2), each line representing the data of a single question for both groups, with the corresponding means and standard deviation values. It includes the p-value of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 2							Group 1										
	S1	**S2**	**S3**	**S4**	**S5**	$\bar{x}$	**σ**	**S1**	**S2**	**S3**	**S4**	**S5**	$\bar{x}$	**σ**	**H1**	**p-both**	**p-right**	**p-left**
Q.1	1	1	1	1	1	1,00	0,00	1	1	1	1	1	1,00	0,00	≠	1,000	1,000	1,000
Q.2	2	1	2	2	2	1,80	0,45	1	1	1	2	2	1,40	0,55	≠	0,524	0,262	0,976
Q.3	2	3	2	3	2	2,40	0,55	3	1	1	4	2	2,20	1,30	≠	0,683	0,341	0,706
Q.4	3	3	3	3	3	3,00	0,00	2	2	3	4	3	2,80	0,84	≠	0,444	0,222	0,861
Q.5	3	4	4	3	3	3,40	0,55	3	1	3	4	3	2,80	1,10	≠	0,643	0,321	0,917
Q.6	3	4	5	3	3	3,60	0,89	4	3	3	4	5	3,80	0,84	≠	0,921	0,778	0,460
Q.7	4	2	3	2	4	3,00	1,00	3	5	3	3	2	3,20	1,10	≠	0,810	0,643	0,405
Q.8	3	4	4	3	4	3,60	0,55	4	5	4	3	4	4,00	0,71	≠	0,643	0,917	0,321
Q.9	2	1	4	3	2	2,40	1,14	1	5	3	3	3	3,00	1,41	≠	0,460	0,802	0,230
Q.10	2	3	4	3	3	3,00	0,71	2	2	3	3	4	2,80	0,84	≠	0,881	0,441	0,798

Table C-2: Pre-experiment questionnaire A results for Group 1 Baseline (G1) and Experimental Group 3 (G3), each line representing the data of a single question for both groups, with the corresponding means and standard deviation values. It includes the p-value of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 3							Group 1										
	S1	**S2**	**S3**	**S4**	**S5**	$\bar{x}$	**σ**	**S1**	**S2**	**S3**	**S4**	**S5**	$\bar{x}$	**σ**	**H1**	**p-both**	**p-right**	**p-left**
Q.1	1	1	1	1	1	1,00	0,00	1	1	1	1	1	1,00	0,00	≠	1,000	1,000	1,000
Q.2	1	1	1	2	1	1,20	0,45	1	1	1	2	2	1,40	0,55	≠	1,000	0,917	0,500
Q.3	3	4	1	2	1	2,20	1,30	3	1	1	4	2	2,20	1,30	≠	1,000	0,595	0,595
Q.4	3	3	3	2	3	2,80	0,45	2	2	3	4	3	2,80	0,84	≠	1,000	0,500	0,679
Q.5	4	4	4	3	4	3,80	0,45	3	1	3	4	3	2,80	1,10	≠	0,167	0,083	0,996
Q.6	4	5	4	4	3	4,00	0,71	4	3	3	4	5	3,80	0,84	≠	0,881	0,441	0,798
Q.7	3	2	2	2	4	2,60	0,89	3	5	3	3	2	3,20	1,10	≠	0,365	0,881	0,183
Q.8	5	4	4	4	3	4,00	0,71	4	5	4	3	4	4,00	0,71	≠	1,000	0,683	0,683
Q.9	4	1	3	3	4	3,00	1,22	1	5	3	3	3	3,00	1,41	≠	1,000	0,500	0,619
Q.10	3	3	3	3	3	3,00	0,00	2	2	3	3	4	2,80	0,84	≠	0,444	0,222	0,861

D. Appendix: Experiment Main Document

The following is a copy of the experiment document handed to the subjects of Group 1 Baseline (G1), Experimental Groups 2 and 3 (G2 and G3) at the beginning the experiment.

Estudio de técnicas de aprendizajes de frameworks de aplicación web - Septiembre 2014

Identificador del Grupo: _____ Nombre Completo: _____

Hora Inicio: _____ Hora Final: _____

Actividad: Una empresa de software requiere sus servicios para desarrollar una aplicación para administrar la información de diferentes cafeterías alrededor de los estados unidos. Después de la fase de análisis y extracción de requisitos, la compañía estableció las siguientes características:

Iteración 1: la aplicación debe tener un menú inicial con 2 opciones ("Administrar cafeterías" y "contacto"). En la sección de "contacto" se deben mostrar los datos del creador de la aplicación, tales como: nombre, email y teléfono.

Hora a la que termine la iteración 1: ____

Iteración 2: En la sección de administrar cafeterías se debe mostrar toda la información de las cafeterías de la base de datos. Para cada cafetería se debe mostrar: su id, nombre, dirección física, ciudad y estado. **Nota:** se anexa un archivo .sql con la información de las cafeterías, este archivo lo debe cargar en una base de datos.

Hora a la que termine la iteración 2: ____

Iteración 3: En la lista de cafeterías anterior, al lado de cada cafetería, se debe anexar un botón o enlace de "eliminar". El cual al ser presionado eliminará la cafetería seleccionada de la base de datos y mostrará un mensaje diciendo "La cafetería ha sido eliminada exitosamente". **Nota:** trate de no borrar las cafeterías con id 20152 y 20153 ya que son las únicas que tienen baristas asociadas.

Hora a la que termine la iteración 3: ____

Iteración 4: Finalmente en la lista de cafeterías, a cada nombre de cada cafetería se le colocará un enlace, el cual al ser presionado cargara de la base de datos la lista de baristas que están asociadas a esa cafetería. Mostrando para cada barista sus datos tales como: id, nombre, email y teléfono. **Nota:** en el archivo .sql anterior también están las baristas.

Hora a la que termine la iteración 4: ____

Este proyecto debe ser realizado en Codeigniter V 2.2.0 y para facilidad de la actividad no tenga en cuenta aspectos de seguridad.

Al finalizar la actividad por favor comprima todo el proyecto my_app en un .rar y envíelo por correo a dcorreab@unal.edu.co Muchas gracias.

E. Appendix: Sql File

The following is the source code of the .sql file used by the subjects of all groups to create the database corresponding to the experiment.

```
CREATE TABLE IF NOT EXISTS `coffee_store` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) NOT NULL,
  `physical_address` varchar(100) NOT NULL,
  `city` varchar(100) NOT NULL,
  `state` varchar(100) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`,`physical_address`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=45500 ;

INSERT INTO `coffee_store` (`id`, `name`, `physical_address`, `city`, `state`)
VALUES
(20152, 'Sweet Beach Cakery', '2200 E 2nd St Bldg J', 'Gulf Shores', ' AL'),
(20153, 'Heritage House Coffee & Tea', '18 McFarland Blvd', 'Northport', ' AL'),
(20154, 'Tasty Tea', 'Not found', 'Alabaster', ' AL'),
(20155, 'Southern Decadence Desserts', '1956B S University Blvd', 'Mobile', ' AL'),
(20156, 'Florist Plus', '7938 Vaughn Rd', 'Montgomery', ' AL'),
(20157, 'Sonny''s Catfish Cafe'", '815 County Road 61', 'Houston', ' AL'),
(20158, 'Chat A Way Cafe', '4366 Old Shell Rd', 'Mobile', ' AL'),
(20159, 'Starbucks Coffee', '801 20th St S', 'Birmingham', ' AL'),
(20160, 'O''Henry''s Coffee', '2831 18th St S', 'Birmingham', ' AL'),
(20161, 'Red Diamond Coffee & Tea', 'Not found', '', 'ot'),
(20162, 'Starbucks Coffee', '1510 Government St', 'Mobile', ' AL'),
(20163, 'Starbucks Coffee', 'Not found', '', 'ot'),
(20164, 'G N U''s Room', '414 S Gay St', 'Auburn', ' AL'),
(20165, 'Starbucks Coffee', '1015 Memorial Pkwy NW', 'Huntsville', ' AL'),
(20166, 'Sunset Cafe', '203 E Main St', 'Samson', ' AL'),
(20167, 'Cafe On Main', '110 2nd Ave W', 'Oneonta', ' AL'),
(20168, 'Green Acres Cafe South', '8500 1st Ave N', 'Birmingham', ' AL'),
(20169, 'The Daily Perk', '913 N Daleville Ave', 'Daleville', ' AL'),
(20170, 'Starbucks Coffee', '2000 Riverchase Galleria', 'Birmingham', ' AL'),
(20171, 'Starbucks Coffee', '2056 Interstate Dr', 'Opelika', ' AL'),
(20172, 'Seattle''s Best Coffee', '2601 Mamie L Foster', 'Birmingham', ' AL'),
(20173, 'American Coffee House And Company', '22229 Highway 31', 'Flomaton', '
AL'),
(20174, 'Carpe Diem Coffee & Tea Company', '4072 Old Shell Rd', 'Mobile', ' AL'),
```

```
(20175, 'Bay Breeze Cafe', '50 S Church St # D', 'Fairhope', ' AL'),
(20176, 'Starbucks Coffee', '3255 Airport Blvd', 'Mobile', ' AL'),
(20177, 'Starbucks Coffee', '1650 US Highway 98', 'Daphne', ' AL'),
(20178, 'Bee Hive', '11 W Claiborne St', 'Monroeville', ' AL'),
(20179, 'Tomek''s', '2320 2nd Ave N', 'Birmingham', ' AL'),
(20180, 'Chelsea Coffee House', '109 Foothills Pkwy', 'Chelsea', ' AL'),
(20181, 'Mokas Coffee House Inc', '1204 Shelton Beach Rd Ste 1', 'Saraland', '
AL');

CREATE TABLE IF NOT EXISTS `barista` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `phone` varchar(100) NOT NULL,
  `coffee_store` bigint(20) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `coffee_store` (`coffee_store`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;

INSERT INTO `barista` (`id`, `name`, `email`, `phone`, `coffee_store`) VALUES
(1, 'Sara Holmes', 'sara@test.com', '555-55-554', 20152),
(2, 'Nick Gomez', 'nick@test.com', '555-55-553', 20152),
(3, 'Patrick Viera', 'patrick@test.com', '555-55-51', 20152),
(4, 'John Smith', 'john@test.com', '555-55-52', 20153);

ALTER TABLE `barista`
  ADD CONSTRAINT `barista_ibfk_1` FOREIGN KEY (`coffee_store`)
REFERENCES `coffee_store` (`id`);
```

F. Appendix: Post-questionnaire

The following is a copy of the anonymous post-questionnaire handed to the subjects of Group 1 Baseline (G1), Experimental Groups 2 and 3 (G2 and G3) at the end the experiment.

Estudio de técnicas de aprendizajes de frameworks de aplicación web

Septiembre 2014

Post-cuestionario:

Gracias por participar de este experimento, ahora te invitamos a participar de este cuestionario que no te quitará más de 5 minutos.

Para cada pregunta existen 5 opciones:

1. fuertemente en desacuerdo
2. en desacuerdo
3. no estoy ni de acuerdo ni en desacuerdo
4. estoy de acuerdo
5. estoy fuertemente de acuerdo

Identificador del Grupo: _____

Preguntas:

Factores externos:

	1	2	3	4	5
El ambiente del trabajo del experimento fue intimidante.					
Disfrute programar y tratar de desarrollar las aplicaciones durante el experimento.					
Me sentí distraído por otros compañeros durante el experimento.					
Disfrute que el desarrollo del experimento fuera individual.					

Promedio de satisfacción

	1	2	3	4	5
Considero que el tiempo estipulado para el desarrollo del experimento fue el adecuado					
Considero que la documentación disponible fue suficiente para desarrollar el experimento					
Sentí la necesidad de acceder a documentación adicional sobre el uso del framework					
Sin tener en cuenta mi experiencia, considero que las herramientas y documentación disponible retrasaron mi trabajo considerablemente					

Proceso de desarrollo

Se me dificulto encontrar como usar el framework para completar…

	1	2	3	4	5
La iteración 1					
La iteración 2					
La iteración 3					
La iteración 4					

Sobre Codeigniter

	1	2	3	4	5
En las url después de index.php/ va el nombre de la función del controlador que se desea cargar					
Para la creación de formularios Codeigniter posee una librería que se recomienda cargar automáticamente desde el archivo config.php					
Todos los modelos extienden de la clase CI_Model					
Codeigniter me permite conectarme simultáneamente a múltiples bases de datos					
La única forma de enviar datos de un controlador a una vista es mediante la sentencia $this->load->vars($data);					
No puedo ubicar modelos dentro de subcarpetas de la clase model					
Puedo conectar un modelo a una base de datos diferente a la por defecto					
Para definir el controlador por defecto accedo a controller_default.php					
La función $this->db->where me permite filtrar datos que se extraen de la base de datos					
En las vistas no se utiliza la combinación de lenguajes php y html					

Si deseas dejar algún comentario o sugerencia sobre el experimento utiliza el siguiente espacio

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

G. Appendix: Post-questionnaire answers

Table G-1: Post-experiment questionnaire results for Baseline (G1) and Experimental Group 2 (G2), each line representing the data of a single question for both groups, with corresponding means and standard deviation values. It includes the values of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 2							Group 1										
	S1	**S2**	**S3**	**S4**	**S5**	$\bar{x}$	**σ**	**S1**	**S2**	**S3**	**S4**	**S5**	$\bar{x}$	**σ**	**H1**	**p-both**	**p-right**	**p-left**
EF.1	2	1	2	5	2	2,40	1,52	4	2	1	1	2	2,00	1,22	≠	0,643	0,321	0,802
EF.2	5	5	5	5	5	5,00	0,00	3	3	4	5	5	4,00	1,00	≠	0,167	0,083	1,000
EF.3	1	1	2	1	3	1,60	0,89	3	1	3	1	3	2,20	1,10	≠	0,524	0,897	0,262
OS.1	4	3	3	4	5	3,80	0,84	2	3	2	2	4	2,60	0,89	>	0,056	0,028	0,996
OS.2	3	5	4	4	4	4,00	0,71	2	3	2	3	2	2,40	0,55	>	0,024	0,012	1,000
OS.3	3	1	3	3	3	2,60	0,89	4	5	5	5	4	4,60	0,55	<	0,008	1,000	0,004
OS.4	3	1	2	2	5	2,60	1,52	5	5	4	4	4	4,40	0,55	<	0,071	0,976	0,036
DP.1	2	1	4	1	1	1,80	1,30	3	5*	4	3	1	3,20	1,48	<	0,206	0,929	0,103
DP.2	3	1	2	1	1	1,60	0,89	5*	5*	5*	4	3	4,40	0,89	<	0,016	1,000	0,008
DP.3	3	1	2	3	1	2,00	1,00	5*	5*	5*	5*	5*	5,00	0,00	<	0,008	1,000	0,004
DP.4	5*	5*	2	5*	1	3,60	1,95	5*	5*	5*	5*	5*	5,00	0,00	<	0,444	1,000	0,222

* The subject wasn't able to complete the task and the highest value –five– was assigned.

Table G-2: Post-experiment questionnaire results for Baseline (G1) and Experimental Group 3 (G3), each line representing the data of a single question for both groups, with corresponding means and standard deviation values. It includes the values of the non-parametric significance Mann-Whitney-Wilcoxon test.

	Group 3							Group 1										
	S1	**S2**	**S3**	**S4**	**S5**	$\bar{x}$	**σ**	**S1**	**S2**	**S3**	**S4**	**S5**	$\bar{x}$	**σ**	**H1**	**p-both**	**p-right**	**p-left**
EF.1	1	1	2	5	1	2,00	1,73	4	2	1	1	2	2,00	1,22	≠	0,921	0,659	0,460
EF.2	5	5	5	5	4	4,80	0,45	3	3	4	5	5	4,00	1,00	≠	0,286	0,143	0,976
EF.3	2	1	1	1	1	1,20	0,45	3	1	3	1	3	2,20	1,10	≠	0,167	0,976	0,083
OS.1	4	4	3	5	5	4,20	0,84	2	3	2	2	4	2,60	0,89	>	0,056	0,028	0,996
OS.2	5	5	3	5	3	4,20	1,10	2	3	2	3	2	2,40	0,55	>	0,048	0,024	1,000
OS.3	3	1	3	1	3	2,20	1,10	4	5	5	5	4	4,60	0,55	<	0,008	1,000	0,004
OS.4	2	3	3	1	1	2,00	1,00	5	5	4	4	4	4,40	0,55	<	0,008	1,000	0,004
DP.1	1	1	2	1	1	1,20	0,45	3	5*	4	3	1	3,20	1,48	<	0,048	0,996	0,024
DP.2	2	1	5	5*	2	3,00	1,87	5*	5*	5*	4	3	4,40	0,89	<	0,286	0,917	0,143
DP.3	2	1	5*	5*	5*	3,60	1,95	5*	5*	5*	5*	5*	5,00	0,00	<	0,444	1,000	0,222
DP.4	4	5*	5*	5*	5*	4,80	0,45	5*	5*	5*	5*	5*	5,00	0,00	<	1,000	1,000	0,500

* The subject wasn't able to complete the task and the highest value –five– was assigned.

Table G-3: Post-experiment questionnaire framework knowledge items results for all Groups. A value of 1 means the questions was correct, 0 incorrect, * the subject didn't know the answer.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total Correct Answers
G1 S1	1	*	*	*	0	*	*	*	*	1	2
G1 S2	0	*	*	*	1	*	*	*	*	*	1
G1 S3	1	*	1	*	*	*	*	*	0	*	2
G1 S4	1	*	1	1	1	*	*	*	*	1	5
G1 S5	1	*	1	*	1	*	1	0	1	*	5
G2 S1	0	*	1	*	*	1	1	*	1	0	4
G2 S2	0	1	1	*	*	*	1	*	1	1	5
G2 S3	1	*	1	0	*	1	*	*	1	*	4
G2 S4	0	0	1	1	*	*	*	*	1	*	3
G2 S5	0	1	1	1	*	0	1	*	1	1	6
G3 S1	0	*	0	*	*	1	1	1	1	1	5
G3 S2	1	1	1	1	0	1	0	*	1	1	7
G3 S3	1	*	1	1	*	*	*	*	1	*	4
G3 S4	0	1	1	0	1	1	1	*	*	1	6
G3 S5	0	*	1	0	0	1	*	*	1	*	3

H. Appendix: Time Results

Table H-1: Iterations time results. Units in minutes.

	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Group 1 Subject 1	109	-	-	-
Group 1 Subject 2	-	-	-	-
Group 1 Subject 3	102	-	-	-
Group 1 Subject 4	42	62	-	-
Group 1 Subject 5	51	36	-	-
Group 2 Subject 1	29	28	26	-
Group 2 Subject 2	31	41	20	-
Group 2 Subject 3	37	22	32	19
Group 2 Subject 4	27	30	48	-
Group 2 Subject 5	27	15	27	41
Group 3 Subject 1	20	29	19	32
Group 3 Subject 2	31	30	35	-
Group 3 Subject 3	40	60	-	-
Group 3 Subject 4	37	-	-	-
Group 3 Subject 5	28	58	-	-

References

[1] J. An, A. Chaudhuri, and J. S. Foster. “Static typing for Ruby on Rails,” Proceedings of 24th IEEE/ACM International Conference on Automated Software Engineering, Auckland, New Zealand, 2009, pp. 590-594.

[2] D. Correa, C. M. Zapata, and F. Arango, “Learning of web application frameworks components,” IADIS AC, October. 2013, pp. 155-162.

[3] X. Shi, K. Liu, and Y. Li, “Integrated Architecture for Web Application Development Based on Spring Framework and Activiti Engine,” The International Conference on E-Technologies and Business on the Web (EBW2013), The Society of Digital Information and Wireless Communication, May. 2013, pp. 52-56.

[4] J. Weinberger, P. Saxena, D. Akhawe, and M. Finifter, “A systematic analysis of xss sanitization in web application frameworks,” Computer Security–ESORICS 2011, Springer Berlin Heidelberg, pp. 150-171, 2011.

[5] I. Sommerville, “Ingeniería del software,” Pearson Educación, 2005.

[6] N. Flores and A. Aguiar, “Understanding Frameworks Collaboratively: Tool Requirements,” International Journal on Advances in Software, vol. 3(1 and 2), 2010, pp. 114-135.

[7] D. Hou, “Investigating the effects of framework design knowledge in example-based framework learning,” Proceedings of 24th IEEE International Conference on Software Maintenance, Beijing, China, 2008, pp. 37-46.

[8] J. An, A. Chaudhuri, and J. S. Foster, “Static typing for Ruby on Rails,” Proceedings of 24th IEEE/ACM International Conference on Automated Software Engineering, Auckland, New Zealand, 2009, pp. 590-594.

[9] J. van Gurp, and J. Bosch, “Design, implementation and evolution of object oriented frameworks: Concepts and guidelines,” Software: Practice and Experience, 31(3), 2001, pp 277-300.

[10] W. G. Halfond, “Automated Checking of Web Application Invocations,” Proceedings of IEEE 23rd International Symposium on Software Reliability Engineering (ISSRE), Dallas, TX, USA, 2012, pp. 111-120.

[11] M. Canales, “A Comparative Study of Rapid Development Frameworks for the Creation of a Language Placement Exam Template,” Master’s Thesis, Texas A&M University, 2010.

[12] P. Wang, “Comparison of Four Popular Java Web Framework Implementations: Struts1. X, WebWork2. 2X, Tapestry4, JSF1. 2,” Master’s Thesis, University of Tampere, 2008.

[13] N. Flores and A. Aguiar, “Patterns for understanding frameworks,” Proceedings of 15th Conference on Pattern Languages of Programs (PLoP), Nashville, TN, USA, 2008, pp. 8.

[14] D. Kirk, M. Roper, and M. Wood, "Identifying and addressing problems in object-oriented framework reuse," Empirical Software Engineering, Vol. 12(3), 2007, pp. 243-274.

[15] R. E. Johnson, “Documenting frameworks using patterns,” ACM Sigplan Notices, vol. 27, no. 10, pp. 63-76, 1992.

[16] F. Shull, F. Lanubile, and V.R. Basili, “Investigating reading techniques for object-oriented framework learning,” IEEE Transactions on Software Engineering, vol. 26(11), pp. 1101-1118, 2000.

[17] G. E. Krasner, and S. T. Pope, "A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80," Journal of Object-Oriented Programming, Vol. 1(3), 1998, pp. 26–49.

[18] K. Jackson, R. Biddle, and E. Temper, “Understanding frameworks through visualisation,” Proceedings of 37th International Conference on Technology of Object-Oriented Languages and Systems, Sydney, Australia, 2000, pp. 304-315.

[19] N. Flores, “Patterns and Tools for Improving Framework Understanding: a Collaborative Approach,” Doctoral dissertation, University of Porto, December 2012.

[20] T. Hug, “Didactics of microlearning: concepts, discourses and examples,” Waxmann Verlag GmbH, Germany, 2007.

[21] G. Kamble, “Aop-Introduced Crosscutting Concerns,” Proceedings of International Symposium on Computing, Communication, and Control (ISCCC), October. 2009, pp. 140-144.

[22] F. Garzoto, D. Schwabe and P. Paolini, "DM-A Model Based Approach to Hypermedia Aplication Design," ACM Transactions on Information System, 11 (1), 1993, pp 1-26.

[23] D. Schwabe, G.Rossi, "Developing Hypermedia Applications using OOHDM," Workshop on Hypermedia Development Process, Methods and Models, Hypertext 98, Pittsburg, USA, 1998.

[24] M. J. Escalona, and N. Koch, "Requirements engineering for web applications-a comparative study," Journal of Web Engineering, vol 2(3), 2004, pp. 193-212.

[25] A. Cicchetti, and D. Di Ruscio, "Decoupling web application concerns through weaving operations," Science of Computer Programming, vol 70(1), 2008, pp. 62-86.

[26] S. Meliá, A. Kraus, & N. Koch, "MDA transformations applied to web application development," Web Engineering, Springer Berlin Heidelberg, 2005, pp. 465-471.

[27] X. Kong, L. Liu, and D. Lowe, “Separation of concerns: a web application architecture framework,” Journal of digital information, vol. 6, no. 2, 2005, pp. 1-8.

[28] G. Sousa, S. Soares, P. Borba, and J. Castro, “Separation of crosscutting concerns from requirements to design: Adapting the use case driven approach,” EA, pp. 93-102, 2004.

[29] I. S. Brito, F. Vieira, A. Moreira, and R. A. Ribeiro, “Handling conflicts in aspectual requirements compositions,” Transactions on aspect-oriented software development III, Springer Berlin Heidelberg, pp. 144-166, 2007.

[30] L. Rosenhainer, “Identifying crosscutting concerns in requirements specifications,” Proceedings of OOPSLA Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, Vancouver, Canada, October. 2004.

[31] T. Elrad, M. Aksit, G. Kiczales, and K. J. Lieberherr, “Discussing aspects of AOP,” Communications of the ACM, vol. 44, no. 10, pp. 33-38, 2001.

[32] D. L. Parnas, “On the Criteria To Be Used in Decomposing Systems into Modules,” Communications of the ACM, vol 15, no. 12, pp. 1053–1058, 1972.

[33] T. Van Gog, and N. Rummel, "Example-based learning: Integrating cognitive and social-cognitive research perspectives," Educational Psychology Review, vol 22(2), 2010, pp. 155-174.

[34] R. K. Atkinson, S. J. Derry, A. Renkl, and D. Wortham, "Learning from examples: Instructional principles from the worked examples research," Review of Educational Research, vol 70, 2000, pp. 181–214.

[35] J. Sweller, "Cognitive load during problem-solving: Effects on learning," Cognitive Science, vol 12, 1998, pp. 257–285.

[36] R. Holmes, and G. C. Murphy, "Using structural context to recommend source code examples," Proceedings of the 27th international conference on Software engineering, ACM, 2005, pp. 117-125.

[37] M. Grechanik, K. M. Conroy, and K. A. Probst, "Finding relevant applications for prototyping," Proceedings of the Fourth International Workshop on Mining Software Repositories, IEEE Computer Society, 2007, pp. 12.

[38] D. Hou, and L. Li, "Obstacles in using frameworks and apis: An exploratory study of programmers' newsgroup discussions. Proceedings of IEEE 19th International Conference on Program Comprehension (ICPC), Kingston, ON, Canada, 2011, pp. 91-100.

[39] C. M. Zapata, A. Gelbukh, F. A. Isaza, "Pre-conceptual schema: A conceptual-graph-like knowledge representation for requirements elicitation," Lecture Notes in Computer Science, Vol. 4293, 2006, pp. 17–27.

[40] C. M. Zapata, G. L. Giraldo, and S. Londoño, "Esquemas preconceptuales ejecutables," Avances en Sistemas e Informática, vol. 8, no. 1, p. 2, 2011.

[41] R. Likert, "A technique for the measurement of attitudes," Archives of Psychology 22 140, 1932, pp. 1–55.

[42] D. Correa, F. Arango, and C.M. Zapata, "Driving the Learning of a Web Application Framework by Using Separation of Concerns," The Ninth International Conference on Internet and Web Applications and Services, ICIW 2014, pp. 76-82.

[43] S. Thummalapenta, and T. Xie, "Parseweb: a programmer assistant for reusing open source code on the web," Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, ACM, 2007, pp. 204-213.

[44] M. Bruch, M. Monperrus, and M. Mezini, "Learning from examples to improve code completion systems," Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ACM, 2009 pp. 213-222.

[45] J. Kim, S. Lee, S. W. Hwang, & S. Kim, "Adding examples into java documents," Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering, 2009, pp. 540-544.

[46] StackOverFlow Question [online] http://stackoverflow.com/questions/15737838/how-to-pass-id-in-controller-from-form-action-using-codeigniter [Last visit: 16 October 2014].

[47] G. Lawton, "Web 2.0 creates security challenges," Computer, vol 40 (10), 2007, pp. 13-16.

[48] P. A. Bruck, L. Motiwalla, and F. Foerster, "Mobile learning with micro-content: a framework and evaluation," Proceedings of the 25th Bled eConference eDependability: Reliable and Trustworthy eStructures, eProcesses, eOperations and eServices for the Future, 2012.

[49] J. Watson, “A Case Study: Developing Learning Objects with an Explicit Learning Design,” Electronic Journal of e-Learning, vol 8 (1), 2010, pp. 41-50.

[50] D. Kovachev, Y. Cao, R. Klamma and M. Jarke, "Learn-as-you-go: new ways of cloud-based micro-learning for the mobile web." Advances in Web-Based Learning-ICWL 2011," Springer Berlin Heidelberg, 2011. pp. 51-61.

[51] Driving Learning Application. [Online]. Available from: http://www.frameworkg.com/dl/micro.php [Last visit: 24 October 2014].

[52] TJ. C. Carver, L. Jaccheri, S. Morasca, and F. Shull, “A checklist for integrating student empirical studies with research and teaching goals,” Empirical Software Engineering, vol 15(1), 2010.

[53] WampServer [online] http://www.wampserver.com/en/

[54] Notepad++ [online] http://notepad-plus-plus.org/

[55] Codeigniter Framework [online] https://ellislab.com/codeigniter

[56] Codeigniter Spanish Official Documentation [online] http://escodeigniter.com/guia_usuario/

[57] Driving Learning Application Meso-Tasks Material. [Online]. Available from: http://www.frameworkg.com/dl/examples.php [Last visit: 24 October 2014].

[58] Driving Learning Application Complete Material. [Online]. Available from: http://www.frameworkg.com/dl/micro.php [Last visit: 24 October 2014].

[59] Yii Framework [online] https://www.yiiframework.com/

[60] Prado Framework [online] https://www.pradosoft.com/

[61] MVC4 Framework [online] https://www.asp.net/mvc/mvc4

[62] Ruby on Rails Framework [online] https://rubyonrails.org/

[63] Cakephp Framework [online] https://cakephp.org/

[64] M. Hollander and D. A. Wolfe, “Nonparametric statistical methods,” Wiley-Interscience, 1999.

[65] J. Plekhanova, “Evaluating web development frameworks: Django, ruby on rails and cakephp,” Institute for Business and Information Technology, 2009.

[66] M. Björemo, and P. Trninić, “Evaluation of web application frameworks-Evaluation of web application frameworks with regards to rapid development,” Master’s Thesis, University of Gothenburg, Göteborg, Sweden, 2009.

[67] J. M. Carroll, P. L. Smith-Kerker, J. R. Ford, and S. A. Mazur-Rimetz, “The minimal manual,” Journal of Human-Computer Interaction, vol 3(2), pp. 123-153, 1987.

[68] M. P. Robillard, “What makes apis hard to learn? answers from developers,” IEEE Software, vol 26(6), pp. 27-34, 2009.

[69] A. Forward and T. C. Lethbridge, “The relevance of software documentation, tools and technologies: a survey,” In Proceedings ACM Symposium on Document Engineering, pp. 26-33, 2002.

[70] J. Nykaza, R. Messinger, F. Boehme, C. L. Norman, M. Mace, and M. Gordon, “What programmers really want: results of a needs assessment for sdk documentation,” In Proceedings International Conference on Computer Documentation, pp. 133-141, 2002.

[71] R. Johnson, “Components, framework, patterns,” SIGSOFT Software, Engineering Notes, vol 22(3), pp. 10-17, 1997.

[72] M. Fayad, D. Schimdt, and R. Johnson, “Building application frameworks,” Wiley Computing Publishing, 1999.

[73] G. Froehlich, H. Hoover, L. Lui, and P. Sorenson, “Hooking into object-oriented application frameworks,” Proceedings of the 19th International Conference on Software Engineering, pp. 491–501, 1997.

[74] S. B. Ho, I. Chai, and C. H. Tan, “Leveraging Framework Documentation Solutions for Intermediate Users in Knowledge Acquisition,” International Journal of Information Science, vol 3(1), pp. 13-23, 2013.

[75] P. Roberts-Morpeth and J. Ellman, “Some security issues for web based frameworks,” Proceedings on the 7th International Symposium in Communication Systems Networks and Digital Signal Processing (CSNDSP), IEEE, pp. 726-731, 2010.

[76] W. K. Robertson, and G. Vigna, “Static Enforcement of Web Application Integrity Through Strong Typing,” In USENIX Security Symposium, pp. 283-298, 2009.

[77] T. Scholte, W. Robertson, D. Balzarotti, and E. Kirda, “An empirical analysis of input validation mechanisms in web applications and languages,” Proceedings of the 27th Annual ACM Symposium on Applied Computing, ACM, pp. 1419-1426, 2012.

[78] K. Jayaraman, G. Lewandowski, P. G. Talaga, and S. J. Chapin, "Enforcing request integrity in web applications," Data and Applications Security and Privacy XXIV, Springer Berlin Heidelberg, pp. 225-240, 2010. Springer Berlin Heidelberg.