Driving the Learning of a Web Aapplication Framework by Using Separation of Concerns

Daniel Correa Botero, Fernando Arango Isaza, Carlos Mario Zapata Jaramillo
Universidad Nacional de Colombia
Medellín, Colombia
Emails: {dcorreab, farango, cmzapata}@unal.edu.co

***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 concern. Commonly, developers follow a guide containing too much information, but in some cases developers only need to learn an incomplete WAF usage. After analyzing some software projects, we define in this paper a list of web application concerns. This list is connected to a list of WAF components, indicating for each concern the specific elements a developer should know for understanding and covering the concern. Such a list helps the developer to drive the WAF learning. We also develop a web application for driving the WAF learning and an example with a real case of driving WAF learning.**

Keywords-Framework learning; WAF; concerns; framework comprehension; WAF components.

I. INTRODUCTION

Developing web systems is a complex, time-consuming, and expensive task that often requires the coordination of efforts across organizational and technical boundaries [1]. 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 [2][3]. Since WAFs are considered crucial for rapid web development [4], several frameworks are available, and this topic is being included in research and development [14][23][24].

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. Several authors have proposed different framework documentation techniques such as: patterns [5], example-based learning [6], cookbooks [7], and visualizations [8]. Some of these techniques are adapted to WAF development. However, they only focus on how to document or show the framework information (architecture, components, classes, relationships, libraries, etc). Currently, when a developer has to use a specific WAF, he/she has to invest considerable effort on understanding it [9]. This problem is due to the big amount of WAF components and the increasing number of documents. Sometimes, developers need to face the reading of hundreds of documentation pages with information they never going to use.

However, in most cases developer learning is primarily influenced by the specific requirements of the application he/she wants to develop. So, developers only need to be concerned on the WAF elements needed to fulfill those requirements. Then, how to drive the WAF learning to be focused on those concerns is an important issue.

In the software development context, a concern is a particular goal, concept, or area of interest [10]. 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. For example: authorization, data storage, internationalization and client-side validation are different types of concerns supporting different kinds of application requirements.

Separation of Concerns (SoC) has been used in multiples software areas during the last years, e.g., requirements specifications [11], framework architectures [1], and aspect-oriented programming [12]. 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 [13].

We use these separation of concerns connected to WAF components [14], giving a specific structure of the elements that a developer should learn for supporting the application requirements. By following such ideas, we develop a new WAF learning technique in which a developer only needs to select the concerns related to his/her development or project, and it will show to him/her the specific components and documentation related. This technique helps developers to save time and to focus on what really they need to learn.

In this paper, we propose a list of 29 basic concerns, and a connection between the list of the concerns and a list of WAF components. Next, we develop a representation of the driving the WAF learning. After that, we develop a simple web application for driving WAF learning, and finally we

 ISBN: 978-1-61208-361-2

develop an example with a real case of driving *Codeigniter* learning.

II. FRAMEWORK UNDERSTANDING

Over the past decade, several documentation techniques have been proposed to support the framework understanding such as: patterns, example-based learning, and cookbooks, among others. However, such techniques are still immature and unused for developing software [2].

Shull et al. [6] show an evaluation of the role examples play in framework reuse. As the main hypotheses, they propose example-based techniques as appropriate to be used by beginning learners instead of hierarchy-based techniques because the latter have a larger learning curve. However, the case study they use is based on a specific example with no patterns, preventing reuse for other frameworks.

Flores and Aguiar [9] present some pattern-based proven solutions to recurrent problems for framework understanding. However, such solutions are top-level basic suggestions.

Jackson et al. [8] support the programmers in understanding the framework code by providing animated visualizations of example programs interacting with the framework. However, a comparison with other methods is not provided.

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. However, they provide too much information, so they slow the framework learning process.

Most of the aforementioned studies are only focused on documenting the framework information—architecture, components, classes, relationships, libraries, etc.—instead of addressing the framework learning for developers.

Flores [25] 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. Besides, DRIVER is still under development and improvement.

In conclusion, several framework documentation techniques have been proposed, but how to address the framework learning is still a developer task. Besides, these techniques are applied to general frameworks, so WAFs are still underspecified.

III. CONCERN LIST

Developers use WAFs for different reasons: developing a software project, acquiring more knowledge, applying for a job position, accessing the training about tools in organizations, etc. 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.

In the first case, CRM system involves a lot of requirements, more than the other applications. It means developer A should learn and read more information than the other developers. We could also recognize application B maybe involves less data persistence and less database effort, and maybe 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.

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. [10].

Some authors have defined different concern lists or methods to define concerns [1][11][15][16], 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 ambiguous elements 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 the last 2 years.

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 I, 29 concerns and we categorize them in different groups.

This concerns list should be used by a developer. 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.

Later on, each concern will be connected to the specific components or elements of a WAF. This generates a personalized learning guide.

 ISBN: 978-1-61208-361-2

TABLE I. LIST OF WEB APPLICATION CONCERNS

#	**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	Internationalization (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 control	The information displayed on your application screens depends on user location (e.g., show a specific app to a 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	Server-side data validation (20)	Security	Your application require validate data (usually additional 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.

 ISBN: 978-1-61208-361-2

IV. CONCERN LIST VS COMPONENT LIST

Several WAF comparison studies show many similarities between them [17][18]. In a previous work, we defined WAF components based on these similarities. We divided the learning process for each component in a set of fundamental tasks, each task details very specifically how components are composed. Furthermore these tasks guide developers in what they should learn in order to learn and use each component [14].

In a real application, concerns are traduced and codified into different components. Commonly, concerns are related to aspect-oriented programming and they are codified into aspects [19]. In object-oriented programming concerns could be traduced into classes and/or components. They also could be traduced in several ways: functions and libraries, among others. It depends on the developer techniques, the tools, or the programming architecture.

TABLE II. WAFS CONCERNS LIST VS WAFS COMPONENTS LIST

Component	*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
	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	--

Component	*Task*	*# of related Concerns*
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
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
Automatic code generator	Identify how to call specific SQL statements	--
	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

 ISBN: 978-1-61208-361-2

Due to the WAF features and taking advantage of WAF components separations, we connected application concerns to WAF components and their tasks. This connection gives the possibility to know for each concern what are the specific components and tasks related to start the personalized learning process.

Table II 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. The main idea is each task as a solution for a specific WAF (it 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.

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 component for crucial elements unknown to him/her (e.g., internationalization, caching, and portability, among others). This means that if he/she implements these elements 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 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. In our work we developed an application capable to register these associations.

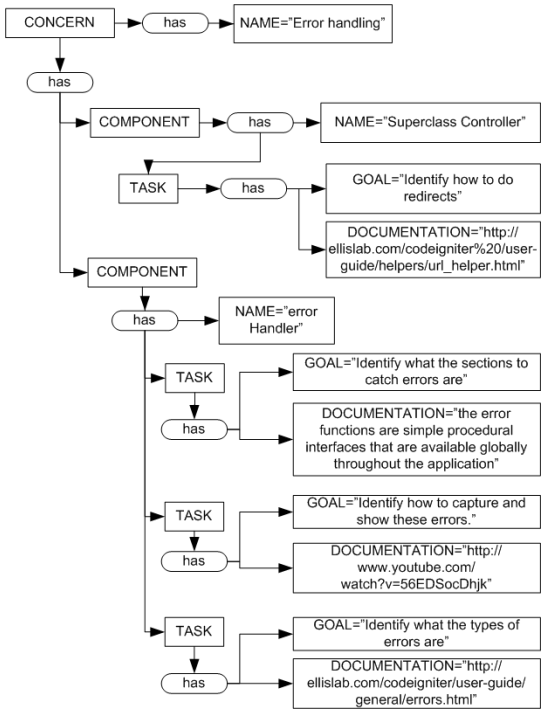

Figure 1. Example of "Error handling" concern, connected to its respectives components and tasks in Codeigniter.

Figure 1 is developed by using an executable pre-conceptual schema [20]. In this figure, we show an example about how concerns, components and learning tasks are connected. If a developer is only interested on capturing and fixing errors, he/she has to analyze and learn 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 tasks he/she has to analyze and learn.

In Figure 2, we summarize the driving of the WAF learning 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 and documentation tasks (previously filled by a senior WAF developer) in order to build the application.

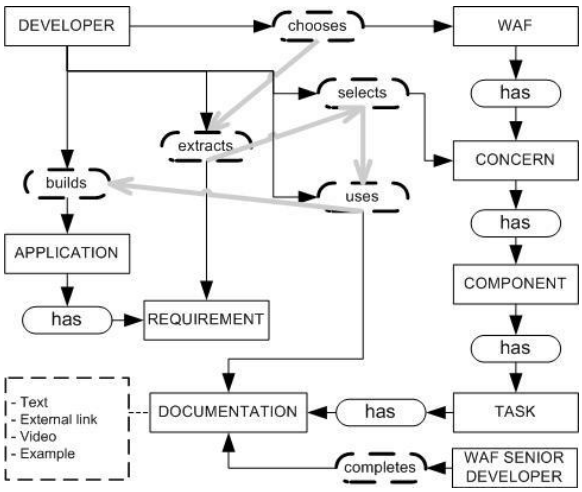

Figure 2. Representing the drive of WAF learning process.

V. CONCERNS SELECTION EXAMPLE

A developer is requested to build an application module by using a new framework. After the requirements elicitation process, a 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 suppose the requested developer should select the concerns listed in Figure 3. Similar to Figure 1, each concern of Figure 3 will be connected to its related components and tasks. Concerns of the Figure 3 support developers as personalized learning guides, i.e., before starting the learning process, developers can discard some documentation unrelated to his/her needs.

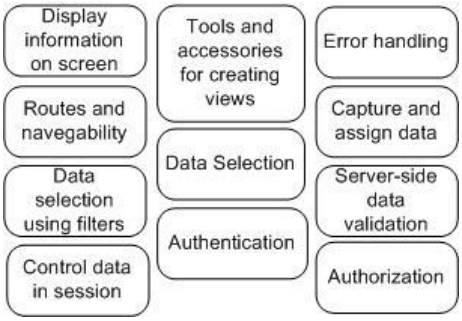

Figure 3. Example of concerns selection.

VI. DL APPLICATION

Tasks documentation is a WAF senior developer job. We developed a driving learning (DL) application [26] with the aim of having this documentation available online and only documenting tasks once in a specific WAF. This is a simple web application which authorizes developers to build a personalized learning guide (see Figure 4).

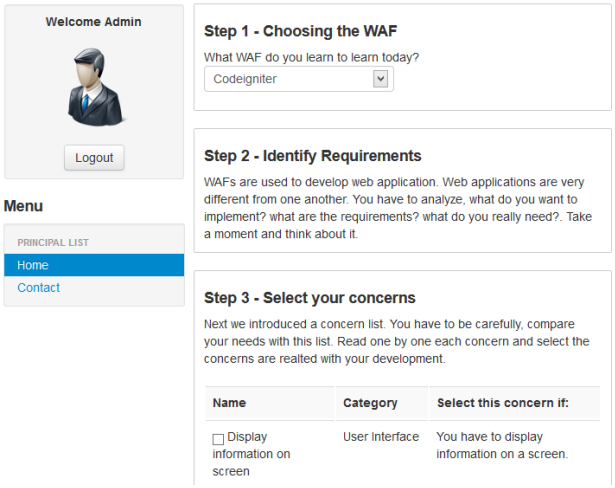

Figure 4. Home of DL application.

Figure 5 shows a real case on *Codeigniter*. The developer complete the previously form and only chose "Error handling" concern. The applications show him/her the solution to this concern with the specific components and tasks he/she has to develop. This guide also allows WAF senior developer to create some notes for each task in order to better complete the information.

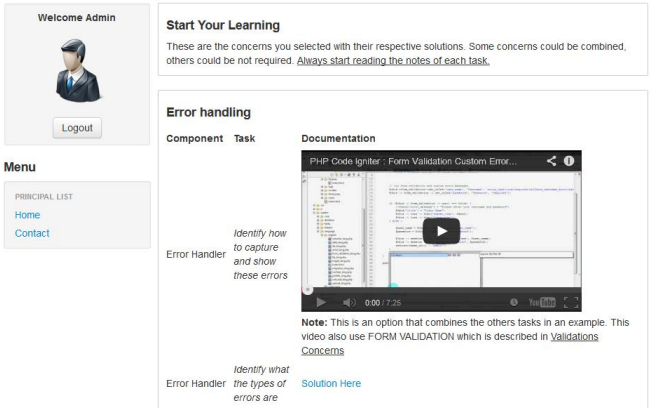

Figure 5. An example of a personalized learning guide on DL application.

Some advantages of this approach are:

- Developers will find a way to guide their learning focusing only in what is concerned to them.
- By learning the basic concerns—first concerns—developers has to understand the framework fundamentals as architecture, folder layout, and basic syntax. E.g., 'display information on screen' concern will give developer the framework fundamental elements.
- Material should be developed by WAF senior developers which guarantee no time wasting on deprecated or wrong internet solutions.
- Future work will connect concerns with a specific example gluing together the components and tasks. As a bonus, exercises provide a source of code reuse —e.g., 'display information on screen' concern connected with 'hello world' example portraits the framework architecture and a base code for all apps—.

VII. CONCLUSIONS

WAF learning is an important issue. Nowadays, WAF learners have to face hundreds of documentation pages and web documents, but they really need to read and follow some parts of the documentation. The main objective of these developers is to build web applications which have different requirements from one to another. By related the documentation to the developer needs, we reduce the amount of documents they have to face, and focus them on what they need. Web application concerns are connected to WAF components giving the possibility to know for each concern what are the specific components and tasks related. This connection is completed by a WAF senior developer; he/she develops all documentation in a specific WAF. Our DL application supports this documentation. In the final step, a WAF learner starts his/her learning process by selecting the concerns related to his/her requirements over DL application and accessing to their personalized learning guide.

 ISBN: 978-1-61208-361-2

VIII. FUTURE WORK

Programmers frequently use a copy-and-paste process to develop their applications [21]. We will improve this learning guide with examples for each concern. A developer has the basic example of each concern and he/she could use it in his/her applications. We will use micro-learning [22] to separate the different steps of WAF learning and finally we will develop a real experimental design to obtain stats and better results.

DL application could also be improved allowing forum discussions and star rating documentation, also increasing the amount of material.

Comparison between different learning techniques like example-based learning, cookbooks, micro-learning and other techniques could be developed.

IX. REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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

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

[15] 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.

[16] 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.

[17] P. Wang, "Comparison of Four Popular Java Web Framework Implementations: Struts1. X, WebWork2. 2X, Tapestry4, JSF1. 2," Doctoral dissertation, Master's Thesis, University of Tampere, 2008.

[18] M. Canales, "A Comparative Study of Rapid Development Frameworks for the Creation of a Language Placement Exam Template," Doctoral dissertation, Texas A&M University, 2010.

[19] M. Marin, A. van Deursen, L. Moonen, and R. van der Rijst, "An integrated crosscutting concern migration strategy and its semi-automated application to JHotDraw," Automated Software Engineering, vol. 16, no. 2, pp. 323-356, 2009.

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

[21] M. Kim, V. Sazawal, D. Notkin, and G. MurphyKim, "An empirical study of code clone genealogies," ACM SIGSOFT Software Engineering Notes, vol. 30, no. 5, pp. 187-196, 2005.

[22] T. Hug, "Didactics of microlearning: concepts, discourses and examples," Waxmann Verlag GmbH, Germany, 2007.

[23] 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.

[24] 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.

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

[26] Driving Learning Application. [Online]. Available from: http://www.frameworkg.com/dl/.

 ISBN: 978-1-61208-361-2