Methodology for Identification and Evaluation of Web Application Performance Oriented Usability Issues

Mihaela Ciugudean
Computer Science Department, Technical University of Cluj-Napoca
Str. G. Baritiu, 28, 400027, Cluj-Napoca
ciugudeanmihaela@yahoo.com

Dorian Gorgan
Computer Science Department, Technical University of Cluj-Napoca
Str. G. Baritiu, 28, 400027, Cluj-Napoca
dorian.gorgan@cs.utcluj.ro

ABSTRACT
This paper aims to illustrate a methodology for identifying and assessing a set of performance issues encountered in a particular web application, with impact on the usability level. Throughout this methodology, several visual techniques are determined and investigated by taking advantage of the functionalities offered by available performance monitoring tools such as JMeter and VisualVM. This way, the major performance concerns and their impact on the web application usability are easily determined. Therefore, by identifying the problematic parts of a system, corrective measures could be taken, particularly designed to operate on the root cause of the problem, thus leading to the targeted objective. Moreover, some performance improvement recommendations are further on presented in order to enhance the overall usability and the user experience, the actual goal of the research and development activities conducted. These performance concerns might be designed and implemented after fixing the critical parts causing usability problems and consequently, maximizing the user satisfaction and comfort when exploring the functionalities offered by a particular web application.

Author Keywords
Interactive software application; usability; performance improvement; JMeter; VisualVM; user experience; visual techniques.

ACM Classification Keywords
H.3.4 Performance evaluation; D.2.2 Design Tools and Techniques; H.5.2 User Interfaces.

General Terms
Visual techniques; Interactive application; Usability; Measurement.

INTRODUCTION
Over the last past decades, the web has evolved into a growing universe of interconnected web pages and applications, offering the end user a wide variety of services and functionalities. With the web development, the interactive web applications have become easily accessible regardless of place and time concerns and without any installation requirements [1]. As a consequence, end users have begun to favor web applications over traditional desktop applications. On the other hand, the rapidly increasing mobile application market has shown that web applications can be easily installed on devices.

In particular, the web enables marketers to get in touch with visitors of their websites or applications and start communicating with them. Furthermore, the web might be perceived as an excellent sales channel for millions of applications, either large or small.

More precisely, end users take advantage of the wide variety of tools and services offered by web applications, by means of web browsers. Therefore, the customers have the opportunity to access, retrieve and interact with data and content located on different software applications available over the web, as stated by Mohler et al. [2].

However, due to the fast growth of the web, an intensive competition has been created among web applications. According to Stoneham and Dastbaz [3], the web, perceived as an application platform, has raised the standard for the web applications it provides, establishing a norm for highly-interactive dynamic user interfaces with real time collaborative features. In case a certain web application fails to completely fulfill the customer's needs and gets him unsatisfied, he will start using services offered by a similar website. Therefore, application developers have begun to enhance the accessibility, usability and user experience when providing specific services and products in order to meet end users' demands.

When creating a web application, it is not sufficient to provide an intuitive, self-explanatory user interface that will allow even an unexperienced computer user to perform the desired actions. As specified by Shivakumar [4], designing a consistent and standardized user interface may still not satisfy the customer, determining him to abandon your website and switch to a similar one that completely fulfills his demands. All this frustration and annoyance that might lead a user to discard your application is very often caused by a series of usability problems occurring in the website.

Considered one of the most important quality factors for web applications, usability has been receiving great attention, being recognized as a fundamental property for the success of web applications [5]. As previously said, it is not sufficient to satisfy the functional requirements of a web application in order to ensure its success. According to Purba [6], the ease or difficulty experienced by users of these web applications is largely responsible for determining their success or failure. Consequently, usability evaluations and technologies that support the usability design process have therefore become critical in ensuring the success of web applications.

The scalability should be an important characteristics of the web application. The main requirement on the usability concerns with keeping a high level of usability at any level of scalable application. It is not enough to check the usability level for single user, but to check usability of the application for a great or huge number of users.

The aim of this study is to present the way by which the visual techniques allow to recognize the root cause of the usability issues encountered in a scalable web application, thus allowing for immediate identification of the technical reason. The paper presents a way of highlighting the usability issues and a directly causal link to the correspondent technical issues.

By taking advantage of the various performance indicators when automatically testing a web application, a particularized methodology might be applied in order to remove, or at least reduce, the encountered system's weaknesses. This way, time and effort invested in unsuccessful operations to improve the overall usability of a web application are saved by starting to design and implement a solution centered on the specific issue that acts as a bottleneck in the system. By means of technical and technological solutions implemented to fix the determined root cause, the overall application usability will be enhanced.

The rest of this paper proceeds as follows. In Section 2, state of the art is presented and the contribution of the current work is highlighted. Section 3 details the proposed methodology by presenting and exemplifying each step. Finally, some concluding remarks and recommendations are proposed.

RELATED WORKS

The exponential growth of the internet and of web applications lead to the necessity to evaluate them from a quantitative view point. In the past years, valuable methodologies have been used to evaluate the quality of specific web applications and discover possible issues. More precisely, inspection methods such as Cognitive Walkthrough for the Web (CWW) and Web Design Perspectives (WDP) are proposed by Haralambos and Colette [7] to identify usability problems. According to them, the aforementioned methods are characterized by a high degree of subjectivity in usability evaluations. In order to overcome this drawback, these methods are sometimes replaced by inspection metrics-based ones which are likely to reduce the subjectivity degree, such as WebTango and Web Quality Evaluation Method (WebQEM), [8]. More precisely, Web Tango provides quantitative metrics, based on empirically validated metrics for user interfaces to build predictive models in order to evaluate other user interfaces.

On the other hand, WebQEM performs a quantitative evaluation of usability aspects, aggregating them to obtain usability indicators. As stated by Singh, the purpose of this method is to systematically assess characteristics, sub-characteristics and attributes that influence products' quality, finally yielding to global, partial and elementary quality indicators that can help different stakeholders in understanding and improving the assessed product. According to Olsina and Rossi [9], by implementing the four major technical steps of WebQEM methodology, it can be employed in assessing and comparing quality requirements in the operative phase of web sites and applications as well as in early phases of web development projects. By using the methodology, either absent attributes, absent sub-characteristics, or requirements poorly implemented might be easily discovered. As conclude by Mendes and Mosley [10], WebQEM can be used to assess diverse application domains according to different user views and evaluation goals.

Nowadays, more and more companies are concerned with addressing usability aspects when designing their website. This concern has developed as a consequence of the competitive market in which each organization strives to continuously enlarge the variety of users by providing an intuitive, self-explanatory and accessible website. Usability is an even broader goal than accessibility, which refers to how easily a website can be used, understood, and even accessed by people with disabilities.

Starting with 2015, even Google has started to evaluate the usability of web applications, mostly their mobile version, and announced by email their owners of the poor rated usability results. Moreover, there was included a list of aspects to be addressed in order to overcome these usability concerns.

One example of receiver of such an acknowledgement was WordPress, which begun to assess the concerns presented in the email received from Google and check their behavior in the mobile version of the web application.

A similar case study of a website trying to enhance its usability and accessibility features is The American Foundation for the Blind. Their strategy was to undergo a major redesign in order to improve the site's usability and create a more logical, user-friendly information architecture.

However, these are only two examples of websites dealing with usability and accessibility concerns in order to maximize the user experience, maintain and enlarge the number of customers. Many other web applications have realized that a consistent and intuitive user interface is sometimes not enough to please the customer's needs and therefore, started to analyze and fix their usability issues.

METHODOLOGY FOR IDENTIFYING AND SOLVING USABILITY ISSUES

The purpose of this paper is to briefly describe the methodology used for identifying the usability issues in a specific web application and inferring the technical reason. Therefore, the considered context is the following one: starting from a particular web application, its usability is defined as the system' capability to successfully handle a variable number of concurrent users. This case the usability is evaluated by the reported error percentage for each case scenario. Therefore, the main step is to measure the application's responsiveness-oriented behavior when successively increasing the number of users, by taking advantage of the functionalities offered by JMeter tool] [11]. After concluding the existence of such usability issues, their root cause is to be determined by means of VisualVM [11], performance monitoring tool, thus obtaining accurate information regarding the sources of the problem. The further step is to analyze the situation and propose and implement specific solutions to overcome the usability problem. Finally, by measuring the application's behavior after applying the established solution, the followed methodology is validated and conclusions related to its accuracy may be stated.

Usability

Being one relevant component of web applications' quality, defined as the extent to which a system is able to satisfy its customers in efficiently achieving their goals, usability is recognized as a fundamental property for software applications' success. In the context of software lifecycle, usability is perceived as being relevant to all its stages, not only at the end of the product development.

According to Osina and Rossi [9], a software product quality might be defined in terms of fundamental characteristics (usability, functionality, reliability, efficiency, portability, and maintainability) as defined in the ISO/IEC 9126-1 standard, usability being one of them.

Specifically, for the considered web application, usability metrics are assessed by monitoring the system's capacity to handle a reasonable amount of concurrent usage. More precisely, the Application Performance Monitoring tests are executed in the context of successively increasing the workload expressed as the number of concurrent active users and monitoring the system's behavior. Therefore, the exemplified attribute in usability evaluation is the number of concurrent users that is successively increased and obtained results interpreted from the reported error percentage values.

Use case scenario: Online Scrum tool

To begin with, the web application under analysis represents a system capable of providing a tool used as an agile project organizer. More precisely, the application is intended to offer the functionalities of an online tool which allows the persons involved in the Scrum process to perform it regardless of any inconveniences such as physically distributed teams, unreachable customer or any other factors that may negatively affect the Scrum process and thus the corresponding software project, [13, 14].

In this context, one use case that has been studied mainly consists of the following actions to be undergo by a privileged user, say a Scrum Master: login, add a new developer to the project, delete another developer (that might have left) and logout.

Methodology for identifying and solving usability issues

In order to identify and evaluate the issues encountered in a web application, by means of visual techniques offered by performance monitoring tools such as JMeter and VisualVM, several use case scenarios were considered. Consequently, for these use cases, a series of experiments were conducted and the visual results were analyzed so that to determine the root causes of the usability issues and propose technical solutions to successfully solve them.

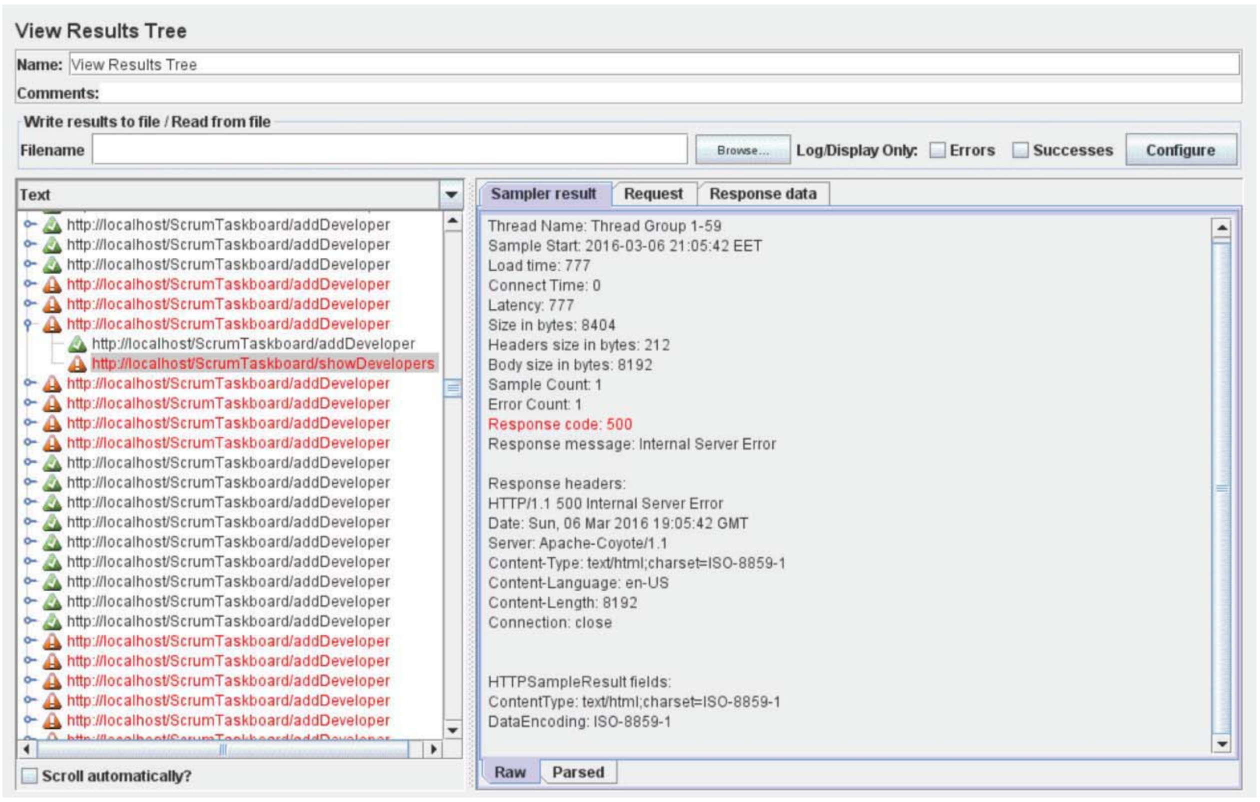
We may define visual techniques as the ensemble of differently shaped graphical representations of the executed testplans that allow evaluation based on various diagnosis indicators while adjusting the variable parameters and performing successive measurements. These visual techniques apply in the context of using dedicated Application Performance Monitoring (APM) tools, such as JMeter and VisualVM, used as a basis in building the methodology for detecting potential performance weaknesses of the system under test and subsequently propose particular solutions to overcome them.

Step 1: Determine the existence of usability problems

In order to assess the non-functional system's characteristics such as usability, availability, scalability, serviceability and other performance oriented metrics, JMeter monitoring tool was firstly used and the reported results analyzed by means of different visual techniques [11].

The aim is to determine potential usability issues in the context of a high workload obtained from a large number of concurrent users that access the web application. The basic idea is that some usability issues are not revealed at low scale of application. The goal is to identify the context in which such issues become detectable. It is difficult and sometimes almost impossible to manually perform all these performance tests, by considering a large number of online users. As a result, these cases were performed by means of the JMeter tool which automatically executes the desired scenarios by simulating the existence of the desired number of users.

The first step of the proposed methodology was to conduct JMeter performance tests, simulating a number of 100 users that simultaneously execute this use case. In the followed context, the considered web application as well as all the system's configuration are the fixed elements, while the number of concurrent users represent the variable elements. Finally, the measurable attributes are the reported error percentage, standard deviation, throughput, number of KB/sec, etc. Specifically for the considered use case, the evaluation is performed by analyzing the reported error by successively increasing the number of concurrent users.



View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename

Browse... Log/Display Only: ☐ Errors ☐ Successes Configure

Text

```
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
    http://localhost/ScrumTaskboard/addDeveloper
    http://localhost/ScrumTaskboard/showDevelopers
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
http://localhost/ScrumTaskboard/addDeveloper
```

☐ Scroll automatically?

Sampler result | Request | Response data

```
Thread Name: Thread Group 1-59
Sample Start: 2016-03-06 21:05:42 EET
Load time: 777
Connect Time: 0
Latency: 777
Size in bytes: 8404
Headers size in bytes: 212
Body size in bytes: 8192
Sample Count: 1
Error Count: 1
Response code: 500
Response message: Internal Server Error

Response headers:
HTTP/1.1 500 Internal Server Error
Date: Sun, 06 Mar 2016 19:05:42 GMT
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Language: en-US
Content-Length: 8192
Connection: close


HTTPSampleResult fields:
ContentType: text/html;charset=ISO-8859-1
DataEncoding: ISO-8859-1
```

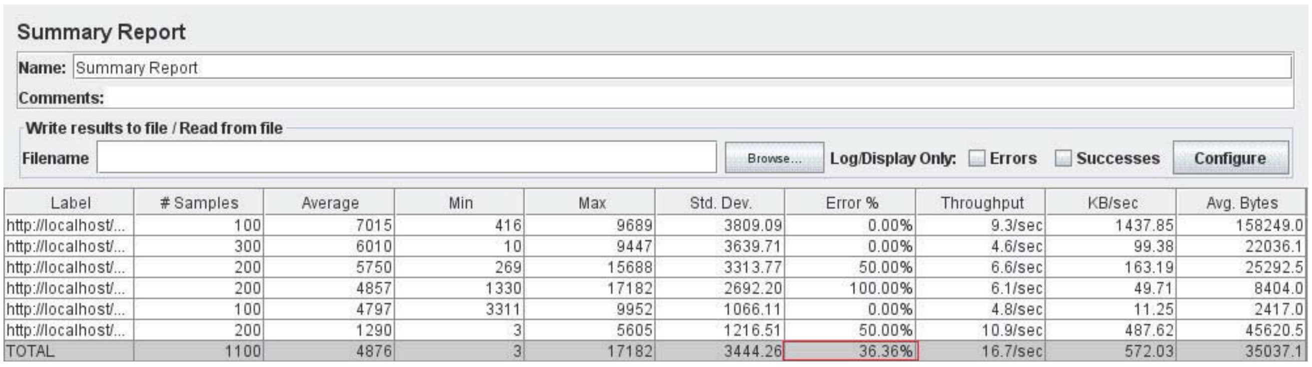
Raw | Parsed

Figure 1. View Results Tree – 100 users.

Finally, the obtained test results are assessed by means of different available listener components which offer visual representation of the output. More precisely, the investigation's objective dictates the number and variety of output listeners that are attached to the executed JMeter test plan.

To begin with, for 100 concurrent threads which simultaneously execute the sample case scenario (login, add a new developer to the project, delete another developer and logout), the application responds with error for some of these 100 HTTP requests. This result might be visualized from the View Results Tree image (which was obtained from JMeter tests) in which the output of each of the 100 user requests to the application (HTTP requests) is displayed, as well as relevant information for all of them (Figure 1). The precise error percentage that was reported for this case scenario is 36.36%, as illustrated in the Summary Report (Figure 2).

As the particular web application is intended to serve a large number of concurrent online users, it has to successfully respond to all of them, or to as much as possible. Therefore, the purpose of this step was to assess the system's behavior in the context of simulating a large users' number. If these results seem worrying for a number of 100 users concurrently accessing the system, the application performs even worse when increasing the number of users to 300. This outcome was determined by performing the same JMeter performance tests with an increased number of 300 users and evaluating the results. The inspected View Results Tree illustrated more HTTP



Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename

Browse... Log/Display Only: ☐ Errors ☐ Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
http://localhost/...	100	7015	416	9689	3809.09	0.00%	9.3/sec	1437.85	158249.0
http://localhost/...	300	6010	10	9447	3639.71	0.00%	4.6/sec	99.38	22036.1
http://localhost/...	200	5750	269	15688	3313.77	50.00%	6.6/sec	163.19	25292.5
http://localhost/...	200	4857	1330	17182	2692.20	100.00%	6.1/sec	49.71	8404.0
http://localhost/...	100	4797	3311	9952	1066.11	0.00%	4.8/sec	11.25	2417.0
http://localhost/...	200	1290	3	5605	1216.51	50.00%	10.9/sec	487.62	45620.5
TOTAL	1100	4876	3	17182	3444.26	36.36%	16.7/sec	572.03	35037.1

Figure 2. Summary report - 100 users.

request that were not handled by the application as compared to the case of 100 concurrent users. Moreover, the reported error percentage was 54.55%, which denotes the system's disability to handle a reasonable usage workload.

Furthermore, in case we increase the users' number, the error percentage increases as well, leading to almost an unresponsive application. The graphic below visually depicts the relationship among the number of concurrent users and the measured error percentage (Figure 3).

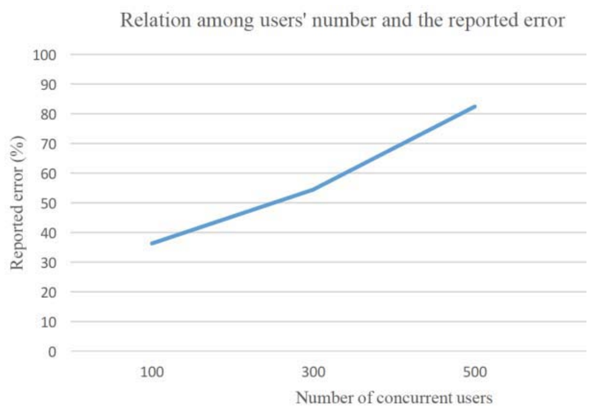


Figure 3. Reported error for different numbers of users.

Consequently, by means of these visual techniques and of the proposed methodology, the system's behavior might be evaluated by subsequently changing the number of users concurrently accessing the application. As bottom line of this first step, we may conclude the erroneous results obtained for a large number of customers, thus leading to poor serviceability and customer experience when taking advantage of the system's functionalities.

Step 2: Determine the root cause of the usability issues

The next step in the proposed methodology is to successfully determine the root cause of the encountered responsiveness-oriented performance issues so that to further on establish concrete solutions to be implemented in order to increase the usability and overall performance of the application. To accomplish this, the simulated test cases executed by taking advantage of the functionalities offered by JMeter, were associated by the ones provided by VisualVM tool [12].

VisualVM as an application performance monitoring tool which freely comes when installing Java JDK, offers several monitoring options, among which the profiling and sampling ones are of interest for our research. The profiler and sampler tools accessible from the VisualVM's IDE are available for both CPU and memory load as performance indicators.

By instrumenting the web application under test, profiling adds a constant amount of extra execution time to every single method call. Sometimes, this results in adding large amount of time to the execution which may even last hours. VisualVM's sampler works by taking a dump of all of the threads of execution on a fairly regular basis, and uses this to determine how much CPU time each method spends. Usually, sampling takes a constant amount of time each second to record stack traces for each thread, thus only adding 5 – 10 minutes of execution time in total, while still succeeding to provide information regarding the source causes of potential performance vulnerabilities. However, sampling has the drawback that the number of invocations recorded is not necessarily accurate, since a short method could easily start and finish between stack dumps. Therefore, the recommendation is to use sampling for recording the amount of CPU time or Memory consumption, rather than the number of invocations, thus having the chance to identify performance problems, much faster than the standard profiler.

Specifically for the scope of our research, only the CPU diagnosis indicator's results were analyzed and, through a series of visual techniques, the system's bottlenecks were identified and further on, specific solutions proposed so that to overcome the performance problems.

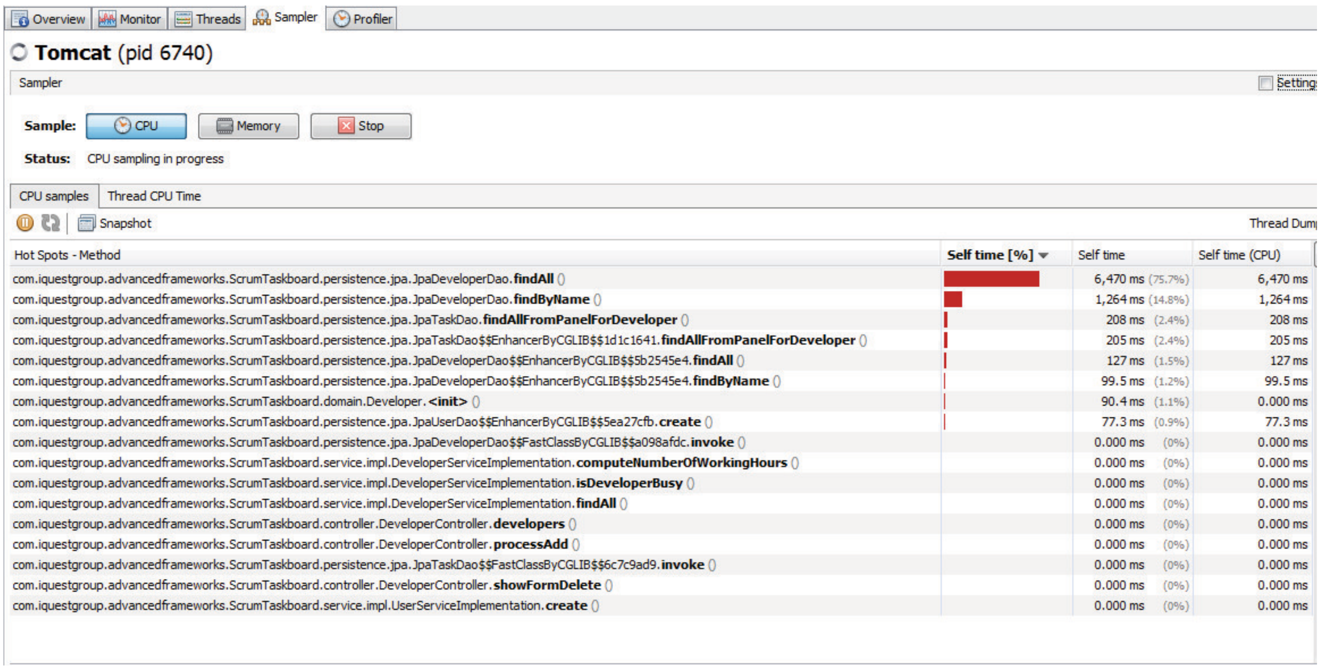
As a result, for the presented use case scenario (login, add a new developer to the project, delete another developer and logout), VisualVM tool offered the ability to perform profiling and sampling activities in order to establish the main bottleneck(s) of the system. Figure 4 presents the CPU usage information for sampling of the considered use case.

These two images offer information related to the bottleneck point of the application, by listing the exact functionalities that consume most of CPU's time. Consequently, the association between the JMeter and VisualVM tools, provide accurate information regarding the precise source of failure for the considered web application and enable specific solutions to be designed and implemented.

Step 3: Analyze results and propose solutions to be implemented in order to overcome the usability issues

By analyzing the results obtained from VisualVM, visual techniques might be used to determine the root cause of the usability issues. Furthermore, from the two images (obtained after experiencing sampling and profiler activities), it can be determined that most of the CPU time is spent on calling functionality at the persistence layer. This outcome is provided by the database method call (findAll(), findByName(), findAllFromPanelForDevelo-per(), …) that consume the most CPU when the considered use case scenario is performed. Therefore, the main CPU bottleneck is caused by methods from the persistence layer (when querying the database) and, thus, further improvements should be implemented at the database level.

Consequently, by means of the visual techniques experienced, the root cause of the encountered usability problems were briefly identified and, further on technical solutions might be proposed and corresponding approached implemented, thus enhancing the overall system's performance.



Status: CPU sampling in progress

Hot Spots - Method	Self time [%]	Self time	Self time (CPU)
com.iquestgroup.advancedframeworks.ScrumTaskboard.persistence.jpa.JpaDeveloperDao.**findAll** ()		6,470 ms (75.7%)	6,470 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.persistence.jpa.JpaDeveloperDao.**findByName** ()		1,264 ms (14.8%)	1,264 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.persistence.jpa.JpaTaskDao.**findAllFromPanelForDeveloper** ()		208 ms (2.4%)	208 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.persistence.jpa.JpaTaskDao$$EnhancerByCGLIB$$1d1c1641.**findAllFromPanelForDeveloper** ()		205 ms (2.4%)	205 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.persistence.jpa.JpaDeveloperDao$$EnhancerByCGLIB$$5b2545e4.**findAll** ()		127 ms (1.5%)	127 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.persistence.jpa.JpaDeveloperDao$$EnhancerByCGLIB$$5b2545e4.**findByName** ()		99.5 ms (1.2%)	99.5 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.domain.Developer.**<init>** ()		90.4 ms (1.1%)	0.000 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.persistence.jpa.JpaUserDao$$EnhancerByCGLIB$$5ea27cfb.**create** ()		77.3 ms (0.9%)	77.3 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.persistence.jpa.JpaDeveloperDao$$FastClassByCGLIB$$a098afdc.**invoke** ()		0.000 ms (0%)	0.000 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.service.impl.DeveloperServiceImplementation.**computeNumberOfWorkingHours** ()		0.000 ms (0%)	0.000 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.service.impl.DeveloperServiceImplementation.**isDeveloperBusy** ()		0.000 ms (0%)	0.000 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.service.impl.DeveloperServiceImplementation.**findAll** ()		0.000 ms (0%)	0.000 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.controller.DeveloperController.**developers** ()		0.000 ms (0%)	0.000 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.controller.DeveloperController.**processAdd** ()		0.000 ms (0%)	0.000 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.persistence.jpa.JpaTaskDao$$FastClassByCGLIB$$6c7c9ad9.**invoke** ()		0.000 ms (0%)	0.000 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.controller.DeveloperController.**showFormDelete** ()		0.000 ms (0%)	0.000 ms
com.iquestgroup.advancedframeworks.ScrumTaskboard.service.impl.UserServiceImplementation.**create** ()		0.000 ms (0%)	0.000 ms

Figure 4. VisualVM Sampling results.

More precisely, for removing the database bottleneck, there are several techniques that may be explored and appropriate solutions implemented. By studying documentation related to database enhancements and methods to be applied on a web application to improve the persistence layer access, a series of rules and guidelines were considered. As one of the main strategy to be applied at persistence layer is adding database indexes where appropriate, this topic was studied in detail and best practices taken into account [15].

According to Singh [16], in order to fix the database bottleneck which determined the application to become unresponsive when a large number of users were trying to access it, the persistence layer strategies were applied where appropriate and the results analyzed after each step. Specifically for the described use case scenario, by means of SQL syntax, slow queries were determined and database indexes implemented in the adequate manner.

Step 4: Evaluating the solutions implemented

By implementing the established solution, the encountered bottleneck at database level was completely removed for the considered use case and the initial error percentages significantly reduced. The developed technical approach was based on an algorithm to determine the database queries which are reported as slow and evaluate the usefulness of adding an index. More precisely, after adding an index on a database table's column, the initial reported error was half reduced. Further on, several analysis were conducted in order to determine the suitability of database indexing for the problematic SQL queries, and indexes added where appropriate. This strategy brought successive performance improvements as a result of reducing serviceability and operability issues (obtained when increasing concurrent usage workload), fact denoted by the minimized reported error percentage (for a large number of concurrent users).

Validation of the implemented methodology

Therefore, by using the aforementioned visual strategies by following the described methodology steps, the root cause of the usability issue was determined and technical solutions were considered and successfully applied. The overall result was an increase in the user experience, satisfaction and response time metrics throughout the system. Consequently, the application will manage not to slow, block or frustrate concurrent customer requests' fulfillment, which determine client loss.

Furthermore, apart from the database bottleneck, the source code was revised and analyzed for the functionalities reported as taking most of CPU's time (in VisualVM's sampler and profiler experiments) and some refactoring strategies proposed. Consequently, the next step in the usability enhancement process was the development of these strategies or even re-implementing of the actual solution (where appropriate) so that to remove possible time or memory consuming algorithms and replace them with a cleaner, cost-effective approach. Both performance improvement methods (the database indexing and the source code refactoring) were considered and applied to other use case scenarios (apart from the one exemplified in this paper) and the system's behavior was definitely enhanced.

To conclude this chapter, the proposed methodology was of great help to determine the existence of usability issues, identify their root cause and allow proposal of particular technical solutions to overcome them. This way, the main operation was performed at the very bottleneck of the system and immediate recovery occurred. As a result, this boosts the overall system performance and offered the desired user experience because of improved usability.

CONCLUSIONS AND FURTHER WORK

As briefly presented in the previous chapter, by following the proposed methodology, the web application's performance problems, oriented towards serviceability, operability and user responsiveness, were spotted and the context clearly defined. Furthermore, by using tools such as JMeter and VisualVM, the obtained results were visually analyzed and the root cause of the issue determined. This way, particular strategies were proposed, centered on fixing the bottleneck, rather than offering a general system improvement (which would only slightly enhance the performance, the root cause not being handled yet).

This methodology proved successful for the considered context and it aimed to enhance the overall system's performance by focusing on the problems identified, i.e. its response time capabilities when increasing the usage workload (defined as the number of concurrent active users). However, attention must be paid to the specifics of a certain system whose performance is to be evaluated and further on improved, because they may differ and therefore, several adjustments have to be applied to the proposed methodology in order to fit particular needs.

Therefore, by applying the described methodologies to fix the encountered problem, system's usability was enhanced, fact clearly proven by the conducted experiments. Not only did the conducted results denote the system's improved performance-centered features by the considerably reduced reported error, but also the methodology validation indicates its efficiency. More precisely, the obvious advantages and strengths of the presented methodology are related to speed in analyzing the system's non-functional requirements from the customer's experience and serviceability viewpoint, as well as accuracy in determining the main bottleneck. The specified speed capabilities refer to the automatization of the testing process by using the JMeter tool which allows to simulate the number of concurrent users, otherwise being almost impossible to assess the system's behavior in this context (as manual testing would be more than cumbersome). As far as the accuracy concerns, the presented methodology offers precise diagnosis of the encountered response time-based user operability problems by identifying their root cause. This way, specific, problem-oriented solutions might be designed and implemented. By focusing on the source of failure, corrective measures are taken at an early stage in the development process.

However, there are still several strategies to be implemented that would obviously boost the entire application performance, and offer a better user experience. More precisely, technical solutions such as adding caching to the developed functionalities and setting up a load balancer are only some examples that involve further study, analysis and implementation effort. All these strategies have to be tested to determine their suitability for our particular web application and establish practical solutions to be developed in order to successfully apply them.

REFERENCES

1. Rossi, G. et al. *Web Engineering: Modelling and Implementing Web Applications*. Springer Science and Business Media, 2007.
2. Mohler, J. L. and Duff, J. M. *Designing Interactive Web Sites*. Delmar/Thomson Learning, 2000.
3. Stoneham, R. and Dastbaz, M. *Building Interactive Web Applications*, Addison-Wesley Longman, Limited, 2006.
4. Shivakumar, S. K. *Architecting High Performing, Scalable and Available Enterprise Web Applications*. Morgan Kaufmann, 2014.
5. Barish, G. *Building Scalable and High-Performance Java Web Applications Using J2EE Technology*. Addison-Wesley Professional, 2002.
6. Purba, S. *Architectures for E-Business Systems: Building the Foundation for Tomorrow's Success*. CRC Press, 2001.
7. Haralambos, M. and Colette, R. Advanced Information System Engineering, *23rd International Conference, CAiSE* 2011, London, (2011).
8. Singh, K. K. *A Quantitative Method for Evaluation of Websites Quality using WebQEM Tool*, Journal of Global Research Computer Science and Technology (JGRCST).
9. Olsina, L. and Rossi, G. Towards Web-site Quantitative Evaluation: defining Quality Characteristics and Attributes. *Proceedings of IV Int'l WebNet Conference*, World Conference on the WWW and Internet, Hawaü, US, 1, (1999), 834-839.
10. Mendes, E. and Mosley, N. *Web Engineering*. Springer Science & Business Media, 2006.
11. Online documentation for JMeter tool: http://jmeter.apache.org/
12. Online documentation for VisualVM tool: https://visualvm.java.net/
13. Schwaber, K. and Sutherland, J. *The Scrum Guide.The Definitive Guide to Scrum: The Rules of the Game*. July, 2013.
14. Rubin, K. S. Essential Scrum: A Practical Guide to the Most Popular Agile Process, Michigan, USA, Pearson Education, 2013.
15. Garner, S. R. Data Warehouse Implementation Strategies: A Mixed Method Analysis of Critical Success Factors, ProQuest, 2007.
16. Singh, H. Data Warehousing: Concepts, Technologies, Implementations, and Management, Prentice Hall PTR, 1998.