Systematic Literature Review on Web Performance Testing

Guilherme Legramante[1], Maicon Bernardino[1], Elder Rodrigues[1], Fábio Basso[1]

[1] Laboratory of Empirical Studies in Software Engineering
Universidade Federal do Pampa (UNIPAMPA)
Caixa Postal 15.064 – 97.546-550 – Alegrete – RS – Brazil

guilhermelegramante@gmail.com,

{bernardino,elderrodrigues,fabiobasso}@unipampa.edu.br

Abstract. *Performance Testing is essential to ensure the quality and scalability of Web applications. A well-defined process may guide Performance Testing Engineer in conducting this task. We intended to enlighten some major inputs related to web performance testing. For this, we have formulated and executed a given protocol, according to the Systematic Literature Review (SLR) protocol in Software Engineering. So, 37 papers were selected/analyzed and we have extracted their most relevant contribution in order to answer our research questions. This analysis enabled us discovering preeminent performance testing profiles/roles, approaches, artifacts, methods, stages or phases and activity flows that have been reported in the literature. We believe that, despite those several studies that mapping performance test context, there are a few remarks in which a clarification might be needed, once there is no well-established process that comprises the whole activities mapped as well as established a relation with other studies. Therefore, this study intends to provide relevant input that one may establish a novel web performance testing process.*

Resumo. *Teste de desempenho é essencial para garantir a qualidade e a escalabilidade de aplicações web. Um processo bem definido pode orientar o Engenheiro de Teste de Desempenho na condução desses testes. Pretendemos investigar os principais conceitos relacionados a Teste de Desempenho. Para isso, executamos uma Revisão Sistemática da Literatura. Assim, 37 artigos foram analisados e suas principais contribuições foram extraídas visando responder às nossas questões de pesquisa. Essa análise nos permitiu descobrir perfis / funções de teste de desempenho proeminentes, abordagens, artefatos, métodos, estágios ou fases e fluxos de atividades que estão relatados na literatura. Apesar de alguns estudos elencarem as entradas e saídas relacionadas ao processo de teste de desempenho, não há um processo bem estabelecido que abranja todas as atividades mapeadas e estabeleça uma relação com outros estudos. Desta forma, este estudo fornece informações relevantes para estabelecer um novo processo de teste de desempenho da web.*

1. Introduction

Web applications need to respond quickly to user actions, once that user engagement is conditioned by the speed in which the system responds to their actions. For instance, a few

seconds of waiting in a determinate task may deflect a purchase in a virtual store, since this delay might make a possible client changing his mind. So, knowing the breaking point of a given system may ensure proper functioning, which allows designing safety mechanisms to support the expected load. Considering the crescent number of systems and Web applications, with a large demand for infrastructure and scalability, we consider that it is necessary to further research that foresees activities related to this demand.

Based on assumptions presented before, it seems a matter of huge importance accessing mechanisms for software performance assessment. So that, by employing a Performance Testing it is possible to plan, execute, monitor, and analyze the results of a system under certain conditions, thereby obtaining the possible expected behaviors of a given software when subjected to these conditions [Bernardino et al. 2016]. In this context, Software Performance Engineering (SPE) [Woodside et al. 2007] can be divided into two general approaches. The former one focuses on early-cycle, by a predictive model-based, *i.e.* performance evaluation and modeling. The latter one adopts a measurement-based approach that involves its late-cycle, *i.e.* performance testing. Considering these assumptions, this research addresses the latter approach, since it enables us to investigate all phases, stages, and activities of performance testing.

For the sake of automating some performance testing tasks, some techniques have been developed. The Capture and Replay (CR) is one of the most used and widespread techniques in performance test automation tools. This technique consists of writing scripts automatically through some system execution functionality. Then, the generated script is executed and the test is performed. Another technique broadly utilized is Model-Based Testing (MBT), in which a model is created, using a specific notation, to generate a test according to planning in the model. MBT may use formal methods to validate the system under test and can be applied either at the hardware or software level. Based on system requirements, test cases are generated based on the models generated [Rodrigues et al. 2015].

Although there are already numerous tools and approaches to performance testing, we assume that information could be better summarized. Discrepancies among defined content by approaches and techniques might hinder the flow of performance testing activities. Based on this, we conducted an SLR on the performance testing area, seeking out comply with to meet this demand. Thus, we selected 37 studies with the purpose of explains the main concepts related to web performance testing. We summarize SLR results in a feature model, followed by a brief explanation of each located input. This provided us an area overview, according to our research questions.

The present paper is organized as follows. In Section 2, we present paper background. Section 3 SLR protocol/execution is presented, followed by SLR results in Section 4. In Section 5, conclusion and future works are discussed.

2. Background

Performance testing is a possibility to plan, execute, monitor, and analyze the results of a system under certain conditions, thereby obtaining the possible expected behaviors of determining software, when subjected to these conditions [Bernardino et al. 2016]. According to Freitas and Vieira [Freitas and Vieira 2014], performance testing is a test that aims to evaluate the performance of the system at a given load scenario. In summary, per-

formance testing provides a load simulation and measurement to detect bottlenecks and the breaking point in which a system crashes under a certain workload.

Woodside [Woodside et al. 2007] defines Software Performance Engineering (SPE) as representing the entire collection of software engineering activities and also related analyses throughout the software development cycle, which are directed to meeting performance requirements. Revealing bottlenecks and achieving improvements in scalability and software performance are some of the main objectives of SPE. In that sense, SPE is classified into two general approaches: predictive model-based and measurement-based. The former one concentrate on the early-cycle and the latter one in the late-cycle of the software development life cycle. Hence, performance testing is associated with a measurement-based approach.

According to Meier *et al* [Meier et al. 2007], a performance testing may be divided into two categories; Load Testing and Stress Testing. A load testing aims to determine a System Under Test (SUT) behavior, which in turn, depicts an application subject a workload. It should be noticed that the load test is conducted to assess if the given system meets specified non-functional requirements. Stress testing places a system under higher-than-expected traffic loads so developers can see how well it works above its expected capacity limits. Moreover, Molyneaux [Molyneaux 2009] includes soak, or scalability, testing, in a way that a soak testing may subject the SUT to a load for a long period, in which some problems dismissed in other categories may become noticeable.

3. Systematic Literature Review Protocol

SLR scope is conducting Performance Testing study area, seeking out for guidelines, taxonomy, process, or frameworks that support activities related to planning, execution, monitoring, and reporting of test results. In this research, we endorsed the protocol proposed by Kitchenham [A. Kitchenham 2007] in SLR. The GQM (Goal, Question, Metric) paradigm [Caldiera and Rombach 1994] usage means to resume the review scope: *For the purpose of* **identify / characterize**, *with respect to* **performance testing processes**, *from the viewpoint of* **performance test engineers and researchers**, *in the context of* **software engineering environment**.

3.1. Research Questions

We assigned the following Research Questions (RQ): **RQ1.** What are the performance testing profiles/roles, artifacts, methods or approaches? **RQ2.** What are the performance testing stages and phases? **RQ3.** What are the performance testing activities, steps or tasks? **RQ4.** What are activity or task flows performed in performance testing?

3.2. Question Structure

Research questions (RQs) are design by means of (PICOC) [Wohlin et al. 2012] criteria, that takes into consideration as follows: **P**opulation: published research on software; **I**ntervention: performance testing; **C**omparison: general comparison of the retrieved processes; **O**utcome: published papers on Performance Testing; **C**ontext: software testing practice and research.

3.3. Search Strategy

To perform the proposed search, we selected the following databases: ACM Digital Library; Engineering Village; IEEE Xplore Digital Library; ScienceDirect; Scopus. These databases were chosen because they stored the main publications in the computer science field and they also offered a web-based search engine. Hence, we elaborated search strings according to each database particularity. The generic string that was used to derive the other strings is shown in Table 1.

Table 1. Generic Search String.

`(process OR framework OR method OR approach OR guideline OR taxonomy OR ontology) AND (web) AND ((performance OR load OR stress OR workload) AND (test OR testing)) AND (stage OR phase OR activity)`

3.4. Selection Criteria and Quality Assessment Criteria

Inclusion criteria and exclusion criteria were defined and applied in order to filter in the initial search. Besides, for a study to be included, it must satisfy at least one inclusion criterion and at least one exclusion criterion as a means to excludes the study from our analysis. To evaluate selected studies' relevance and also answering some research questions, we used quality assessment criteria. The quality assessment criteria are featured which may be exploited in two stages: the first stage being the individual evaluation of each researcher, to reduce bias probability; the second stage is where researchers should reach a consensual note about publications in a "divergent state" in a quality measurement grade. Due to space limitation, we provide a set of documents in an online repository that including the list of inclusion and exclusion criteria; quality assessment criteria, and; data extraction strategy (`http://bit.ly/2wKwUPp`).

3.5. Selection Process

(1) **Pilot Search Strategy:** In order to verify the quality of the proposed search string, the approach called Search-Based String Generation (SBSG) [Souza et al. 2018] was applied. The approach is based on precision and sensitivity indexes calculation. The precision is the ability to identify the amount of *irrelevant* studies, while the sensibility is a measure to identify all of the *relevant* studies. When precision is zero, no irrelevant study is detected. This approach applies an Artificial Intelligence technique through the Hill-Climbing algorithm suggested by Russell [Russell and Norvig 2016], which allows the measurement of precision and sensitivity indexes for a set of keywords and an initial set of selected papers. The proposed *string* was submitted on the basis of 8 (eight) pre-selected studies. Thus, the achieved results were 11.27% precision and 79.49% sensitivity. (2) **Search Databases:** The strings were generated using selected terms and synonyms and were run in the selected databases, resulting in an initial aggregation of studies; (3) **Removal of Duplicates:** The results of initial selection were filtered-out for duplicated entries; (4) **Selection Studies:** In this step, we read separately the title and the abstract (reading the introduction and conclusion when necessary) of each study. Here, we decided to select or reject an article following defined inclusion and exclusion criteria; (5) **Quality Assessment:** The selected studies from inclusion and exclusion criteria application were

submitted to quality assessment criteria; (6) **Data Extraction:** To answer to RQs, the selected/classified studies were obtained and relevant data were extracted using a form.

Our initial selection was conducted in May 2019, on ACM Digital Library, Engineering Village, IEEE Explore, Science Direct, and Scopus and provided close to 1328 results. After filtering out duplicate entries, the number of results was reduced to 1081. The number of duplicate entries was quite large and this might be attributed to papers being revised from conferences publications into journal articles, being extended and submitted in later conferences, and overlapping results from databases. After separately applying inclusion and exclusion criteria fifty-two (52) studies remained. Finally, the quality assessment reduced the number of results was reduced to thirty-seven (37) papers.

4. Results and Discussion

This section presents the SLR results, in which thirty-seven (37) studies are discussed to respond to defined research questions. Figure 1 provides us an overview of obtained results in the form of a feature model. Nodes Test Plan, Model, Planning and Analysis are of optional nature, *e.g.*, an approach that does not use a model as an artifact, this is not required.

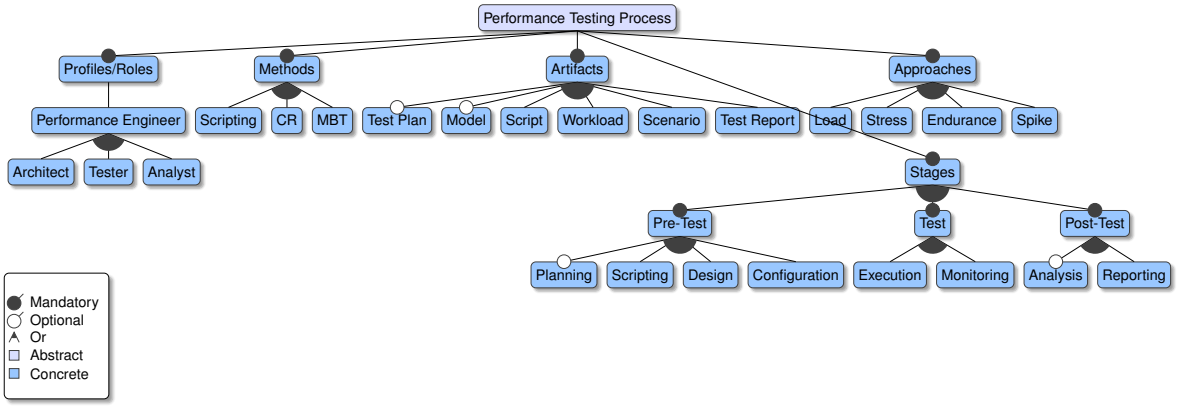

Figure 1. Overview of web performance testing process.

4.1. RQ1. What are the performance testing profiles/roles, artifacts, methods or approaches?

4.1.1. Profiles/Roles

After to analyze selected studies, we identified the following four (4) profiles/roles: (1) **Performance Engineer:** The performance engineer must have the knowledge to support all stages, phases, and activities of the performance test. This role can be specialized in other roles (Performance Architect, Performance Tester, and Performance Analyst). Some papers make reference directly or indirectly to this role [Subraya 2006] [Xu et al. 2014] [Van Der Ster et al. 2011]. (2) **Performance Architect:** This role is involved within Design and Configuration Phases and it must make a connecting bridge between early phases and testing execution. A Performance Architect must-have skills to make design and configuration activities. The term ”Performance Architect” is reported in Subraya [Subraya 2006] paper. (3) **Performance Tester:** This role is directly related

to the testing execution phase. A Performance Tester is the one who should "operate" performance testing, making use of available tools for performing this activity. A few papers bring this role within another nomenclature as User and Developer [Tselikis et al. 2007] [Pfau et al. 2017]. We merged these terms in Performance Tester, once we believe to be more suitable for this context. (4) **Performance Analyst:** Performance Analyst has participation in early and late performance testing phases. He is responsible for initial testing planning and documentation, providing input to subsequent phases, design, and configuration. This role is also present after testing execution, on this account, it is employed in the analysis and reporting phases. This role is not directly reported in the chosen papers. However, Subraya [Subraya 2006] refers to their activities, without specific nomenclature.

4.1.2. Artifacts

The most relevant artifacts are presented to support performance testing activities: **Performance Test Plan:** is a document elaborated by a Performance Analyst as a means to, provide support and guiding the team in the whole test activities. In this document general testing features, such as testing type, scope, approach, and the steps to achieve performance testing goals are explained. This artifact is generated in the planning phase and it is reported in some papers that focuses on this phase [Meier et al. 2007] [Freitas and Vieira 2014] [Yin et al. 2008] [Huang et al. 2011]; **Model:** is used as input in technique known as Model-Based Testing. A model is an abstraction of software behavior that enables reuse and facilitates the understanding of the flow of activities performed by the test [Yin et al. 2008]; **Performance Script:** is the main input artifact for running the test. Through it, the test execution flow is defined, since a script is represented by a set of instructions and may be obtained in an automatic or manual manner. In the former, scripts are generated through tools that use a capture and replay mechanisms [Subraya 2006]. On the latter one, in manual form scripts are generated through a programming language code; **Workload:** is responsible for modifying the SUT situation through its different configurations. A workload may vary based on the test approach and it includes a number of users, concurrent active users, data volumes, and transaction volumes, along with the transaction mix. For performance modeling, a workload is associated with an individual scenario [Pfau et al. 2017]; **Performance Scenario:** defines as a set of steps in an application [Meier et al. 2007]. Moreover, a scenario may map a given application context, within a determinate workload for a user profile, it should be modeled on the basis in usage patterns and log files. **Performance Test Report:** reports the data retrieved by test execution. This report must contain test results, organized in a way that allows their interpretation by stakeholders. Meier *et al* [Meier et al. 2007] lists six key components, which are not mandatory, of a technical report: a results graph, a table with single-instance measurements, a workload model, test environment, general observations and a references section.

4.1.3. Methods

Three methods are related to performance testing conducting [Rodrigues et al. 2015]: **Scripting:** This method involves technique support by the manual script where the performance tester writes a set of code statements, which are going to be inputted to a load

generator to providing a workload in a given scenario; **Model-Based Testing:** In this method, a software behavior under test is verified according to their model. It has some advantages such as enabling the application of models for appropriate testing models creation, as well its use in performance testing; **Capture and Replay:** This method consist of recording the execution of the application's functionalities for the generation of test scripts for the later execution of these scripts simulating the execution of the application's functionalities.

4.1.4. Approaches

Subraya [Subraya 2006] presents a set of four (4) performance testing approaches called LESS (Load, Endurance, Stress, Spike): (1) **Load Testing:** A load is a quantitative of users which compete to increase traffic of application. It is useful for determining the break-point and checking when bottlenecks begin to emerge; (2) **Endurance Testing:** An endurance testing is directly related to the reliability of the application. Different test execution times can be set to check the behavior of the application in different scenarios based on the duration of the test. Endurance testing may be to performed on a normal load or on a stress load, but the main focus of this approach is the test duration; (3) **Stress Testing:** A stress testing is similar to the load testing. However, the stress testing aims to check how the application handles in its limit. Therefore, it helps researches to identify the load that the system can handle before breaking down or degrading quickly; (4) **Spike Testing:** A spike testing is conducted to verify application behavior under a surge in a short duration. The application is subjected to a sudden load increase.

4.2. RQ2. What are the performance testing stages and phases?

For our purpose, stages were mapped as being the activities group in the high level, which may have one or more phases. We identified three (3) stages and nine (9) phases in the performance testing context. The stages and phases are as follows:

4.2.1. Pre-Test

This stage comprises previous phases to test execution. The test definition and preparation are prepared in this stage. The Pre-Test stage has four (4) phases: (1) **Planning:** In this phase occurs test definition. Major requirements related to the test are mapped and some factors should be analyzed, such as network and infrastructure environment, business functions related with the performance requirements and everything that may be really relevant to the test; (2) **Scripting:** This phase involves activities that focus on script elaboration, which can be obtained by different means. For instance, supported by models and uses the MBT or CR for a automatic generation or also by means of coding, where scripts are made from specific programming language; (3) **Design:** Using the test specifications defined in the planning phase, performance testing is designed taking into account environments particularities and performance testing goals; (4) **Configuration:** It is the last phase before test execution. In this state, adjusts and setting performance testing are made. Issues like workload type, performance testing type and tool functionalities should be considered as well as infrastructure issues.

4.2.2. Test

After the Pre-Test stage, in this stage occurs the Test stage, moment in which test execution is realized. **Execution:** In this phase workload is generated and the SUT is monitored to obtain inputs that indicate the main bottlenecks and the behavior of the system under this load. In addition to this monitoring, test should provide mechanisms to collect necessary metrics, which were defined in the Pre-Test stage. **Monitoring:** Defined metrics as throughput, response time, hits per second must be monitored during test execution. This monitoring allows performance testing roles to obtain outputs for subsequent phases, analysis and reporting in post-test stage.

4.2.3. Post-Test

This stage encompasses the phases that postdate Execution, Analysis and Reporting respectively. 1. **Analysis:** In this phase result analysis is conducted, according to the metrics that were collected during test execution. The support by a specific tool is very important in this phase, once manual execution is impracticable. However, this analysis is directly related to the collection of metrics results exposure during the test, not the analysis by the performance engineer in a decision making process; 2. **Reporting:** This phase is the sequence of the analysis phase. In this phase, test results are reported. This report might vary between a detailed and automated report, depending on the tool used, or a report with minor information, so that the performance engineer/architect has the task of interpreting performance testing report results.

It was possible to verify that most of studies addresses test execution. Other issue demonstrated in this table is an evaluation type, achieved by selected studies. Case studies are more recurring in this context, once eleven (11) empirical studies this type were founded, followed by experiments with seven (7) studies that used this approach to assessing the study. Another relevant question to be highlighted is that the majority of studies are not focused on all phases and stages of a performance testing, on the grounds that they focus on some specific phases.

4.3. RQ3. What are the performance testing activities, steps or tasks?

Based on the selected papers, we found one hundred thirty eight (138) performance testing activities, steps or tasks, so it is unfeasible to detail them in this paper In Figure 2 it is possible to identify that assumption, as well as the trend evidenced in previous research questions. There is a greater concentration of activities in the test execution phase, where we identified forty seven (47) activities related to this phase. The other phases have a similar commensurate of activities, ranging from fifteen (15) to twenty two (22) activities, except Scripting and Reporting phase where twelve (12) and five (5) related activities, respectively, were found.

It is relevant to allude that some activities can have differences only in their nomenclature, for the sake of having same objective in practice. It is also worth emphasizing that due to the varied possibilities for a performance test, not necessarily all the mapped activities must be used, as a result of the particularity of each test, a certain group of activities will be executed.

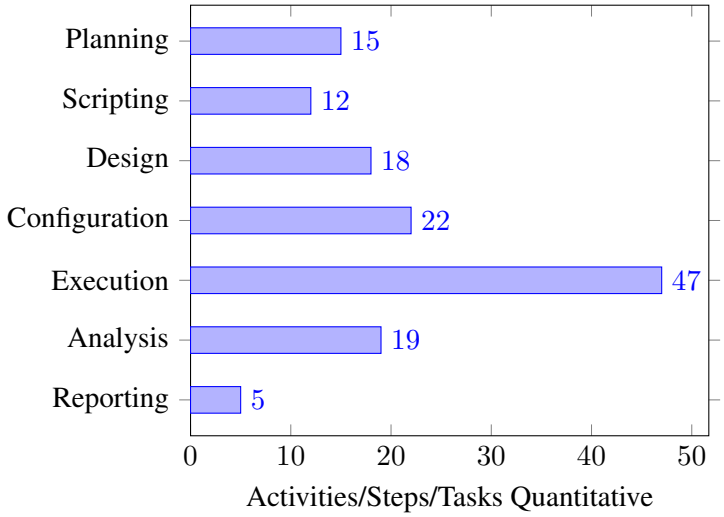

Figure 2. Relation Activities/Steps/Tasks Quantitative X Phases.

4.4. RQ4. What are activities/tasks flow performed in performance testing?

The mapping of the stages, phases and subsequent activities reported in the selected studies allowed us allowed us to understand that the phases can be organized in a circular manner. Performance testing may be thought of as a sequential activity and may be instantiated as many times as necessary. In this flow the sequence starts in the Planning, following the six (6) next phases until completing the cycle. Another reason that motivated us to model the flow in this manner is due to the fact that is the large variety of activities and tasks that do not include all mapped phases, making it possible to understand the sequence of the test independent of the activity described to contemplate the phases in their totality or not.

5. Conclusion and Future Work

Performance Testing is an important area that has a direct influence on application reliability and scalability. Based on this, mapping the main concepts of the area is important when trying to define a generic process in the context of performance testing for Web applications. In this paper, we present the protocol, execution, and results of an SLR for an overview of performance testing for Web applications. Hence, thirty-seven (37) studies were selected and analyzed to obtain subsidies that answered our research questions. We assume that our main contribution was obtained through SLR results, which allowed us to map the main concepts related to the performance testing area, encompassing all its stages and phases. Our results were reported in a textual description of them and by a feature model that encompasses the whole SLR results.

Currently, we are working on the development of a broad and generic web performance testing process, supporting those involved in performance testing activities with a well-defined process that presents roles, phases, and activities with their respective inputs and outputs. This SLR is the initial point of this process, once despite highlighting the main concepts, it is still necessary that they are refined and confronted with the state of practice in the industrial environment. Thus, we have also employed efforts to conduct a survey to this end.

References

A. Kitchenham, B. (2007). *Guidelines for performing Systematic Literature Reviews in software engineering. EBSE Technical Report EBSE-2007-01*.

Bernardino, M., Zorzo, A. F., and Rodrigues, E. M. (2016). Canopus: A domain-specific language for modeling performance testing. In *IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pages 157–167. IEEE.

Caldiera, V. R. B.-G. and Rombach, H. D. (1994). Goal question metric paradigm. *Encyclopedia of software engineering*, 1:528–532.

Freitas, A. and Vieira, R. (2014). An ontology for guiding performance testing. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 400–407. IEEE.

Huang, X., Wang, W., Zhang, W., Wei, J., and Huang, T. (2011). An adaptive performance modeling approach to performance profiling of multi-service web applications. In *Proc. International Computer Software and Applications Conference*, pages 4–13.

Meier, J., Farre, C., Bansode, P., Barber, S., and Rea, D. (2007). *Performance testing guidance for web applications: patterns & practices*. Microsoft press.

Molyneaux, I. (2009). *The art of application performance testing: Help for programmers and quality assurance*. " O'Reilly Media, Inc.".

Pfau, J., Smeddinck, J. D., and Malaka, R. (2017). Automated Game Testing with ICARUS: Intelligent Completion of Adventure Riddles via Unsupervised Solving. In *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play*, pages 153–164, New York, NY, USA. ACM.

Rodrigues, E., Bernardino, M., Costa, L., Zorzo, A., and Oliveira, F. (2015). Pletsperf-a model-based performance testing tool. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, pages 1–8. IEEE.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Souza, F. C., Santos, A., Andrade, S., Durelli, R., Durelli, V., and Oliveira, R. (2018). *Automating Search Strings for Secondary Studies*, chapter 558, pages 839–848. Springer International Publishing.

Subraya, B. M. (2006). *Integrated approach to web performance testing: A practitioner's guide*.

Tselikis, C., Mitropoulos, S., and Douligeris, C. (2007). An evaluation of the middleware's impact on the performance of object oriented distributed systems. *Journal of Systems and Software*, 80(7):1169–1181.

Van Der Ster, D. C., Elmsheuser, J., Garcia, M. U., and Paladin, M. (2011). HammerCloud: A stress testing system for distributed analysis. In *Journal of Physics: Conference Series*, volume 331, Taipei, Taiwan.

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wessln, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.

Woodside, M., Franks, G., and Petriu, D. C. (2007). The future of software performance engineering. In *Future of Software Engineering*, pages 171–187. IEEE.

Xu, X., Jin, H., Wu, S., Tang, L., and Wang, Y. (2014). URMG: Enhanced CBMG-based method for automatically testing web applications in the cloud. *Tsinghua Science and Technology*, 19(1):65–75.

Yin, J., Ming, Z., Xiao, Z., and Wang, H. (2008). A web performance modeling process based on the methodology of learning from data. In *Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS 2008*, pages 1285–1291, Zhang Jia Jie, Hunan, China.