UNIVERSIDADE FEDERAL DO PAMPA

Guilherme Legramante Martins

Towards a Performance Testing Body of Knowledge (PTBOK)

Alegrete
2021

Guilherme Legramante Martins

Towards a Performance Testing Body of Knowledge (PTBOK)

Master Thesis presented as partial requirement for obtaining the degree of Masters of Software Engineering at Federal University of Pampa.

Supervisor: Prof. PhD. Maicon Bernardino da Silveira

Co-supervisor: Prof. PhD. Elder de Macedo Rodrigues

Alegrete
2021

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

```
M386t  Martins, Guilherme Legramante
          Towards a Performance Testing Body of Knowledge (PTBOK) /
       Guilherme Legramante Martins.
          95 p.

          Dissertação(Mestrado)-- Universidade Federal do Pampa,
       MESTRADO EM ENGENHARIA DE SOFTWARE, 2021.
          "Orientação: Maicon Bernardino da Silveira".

          1. Engenharia de Software. 2. Teste de Desempenho. I.
       Título.
```

GUILHERME LEGRAMANTE MARTINS

TOWARDS A PERFORMANCE TESTING BODY OF KNOWLEDGE (PTBOK)

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Software da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia de Software.

Dissertação defendida e aprovada em: 28 de julho de 2021.

Prof. Dr. Avelino Francisco Zorzo

PUCRS

Prof. Dr. Fábio Paulo Basso

Unipampa

ABSTRACT

Due to the growing market demand, web applications need to quickly respond to users' requests, since their engagement may be inclined to fluidity and agility in interactions. For this reason, through performance testing, we may map out the scalability of the application and identify bottlenecks that may affect its performance. There are some studies in academia and industry that provide information for conducting this type of test. However, there is no standardization of procedures, and the information is not organized in a way that facilitates the performance test execution in its complete cycle, from its conception to the generation of the reports. Hence, our research seeks to identify the main inputs, outputs, and processing involved in the performance testing, so that we developed the Performance Testing Body of Knowledge (PTBOK). This body of knowledge aims to assist the preparation and conduction of the performance test. Then, we did a Systematic Literature Review (SLR) that, mapped out the performance testing throughout its life cycle. We found and detailed thirty seven (37) papers, elaborating a feature model from them. We also carried out another empirical study: a survey, which sought to identify (as well as the SLR) data related to the performance testing process from the software industry. After conducting these two studies, we filtered and merged the results, we started the PTBOK creation. For the modeling of the process, we chose the Software & Systems Process Engineering Metamodel (SPEM), as we understand that it is a viable and adequate alternative since it is specifically for modeling software processes. PTBOK is already available and we evaluated it through a survey of performance testing experts.
Key-words: Performance Testing. Software Testing. Software Process.

RESUMO

Devido à crescente demanda do mercado, as aplicações Web precisam responder rapidamente às requisições dos usuários, uma vez que, o engajamento destes usuários pode estar condicionado a uma fluidez e agilidade nas interações. Com base nisso, podemos aplicar um teste de desempenho para mapear a escalabilidade da aplicação e também identificar gargalos que podem afetar o seu desempenho. Há diversos trabalhos no meio acadêmico e também na indústria que fornecem informações para a condução deste tipo de teste. Porém, não há uma padronização de procedimentos e as informações não estão organizadas de forma que facilitem a aplicação de um teste de desempenho em seu ciclo completo, incluindo desde sua concepção até a geração de relatórios. A partir desta demanda, nossa pesquisa busca identificar as principais entradas, saídas e processamento envolvidos no teste de desempenho para que possamos instanciar um processo genérico, o qual denominamos *Performance Testing Body of Knowledge (PTBOK)*, o qual é um corpo de conhecimento em teste de desempenho que visa auxiliar na elaboração e condução do processo de teste de desempenho. Para isso, realizamos uma Revisão Sistemática da Literatura (RSL) que mapeou o teste de desempenho em todo seu ciclo de vida. Encontramos e detalhamos 37 trabalhos e a partir deles criamos um modelo de características. Também realizamos outro estudo empírico, um *Survey*, o qual buscou identificar assim como a RSL, dados relacionados ao processo de teste de desempenho. Após a condução destes dois estudos, filtramos e unimos os resultados dos mesmos e com esse embasamento tanto do meio acadêmico como da indústria, começamos a criação do PTBOK. Para a modelagem do processo escolhemos o meta-modelo SPEM, por entendermos ser uma alternativa viável e adequada, por ser específico para modelagem de processos de software. Avaliamos o PTBOK por meio de um *survey* com especialistas na área de Teste de Desempenho.

Palavras-chave: Teste de Desempenho. Teste de Software. Processo de Software.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

1 INTRODUCTION

Web applications need to quickly respond to users' actions, since the user's engagement is conditioned by the speed at which the application responds to their actions. For instance, a few seconds of waiting in a task may impede a purchase in a virtual store, since this delay might make the client change his mind. Therefore, knowing the application breaking point may ensure proper functioning, which allows designing safety mechanisms to the expected load. Considering the crescent number of Web applications with a large demand for infrastructure and scalability, we consider that it is necessary to develop research that foresees activities related to this demand.

It is a matter of huge importance to access the mechanisms for a software performance evaluation. So that, by employing 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 submitted to those conditions (BERNARDINO; ZORZO; RODRIGUES, 2016).

In this context, Software Performance Engineering (SPE) (WOODSIDE; FRANKS; PETRIU, 2007) can be divided into two general approaches. The former focuses on the early cycle by a predictive model-based, *i.e.*, performance evaluation and modeling. The latter 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 the performance testing.

Performance testing may be performed in several ways, and it has some specific approaches to certain situations such as Load, Endurance, Spike, and Volume Testing. Moreover, some techniques have been developed for the sake of automating some performance testing tasks. The Capture and Replay (CR) (MEMON; SOFFA, 2003) is one of the most used and widespread techniques in performance testing automation tools. This technique consists of writing scripts automatically through some application execution functionality. Then, the generated script is executed, and the test is performed. Another technique broadly utilized is the Model-Based Testing (MBT) (DALAL et al., 1999), in which a model is created using a specific notation, generating test artifacts as planned in the model.

1.1 Motivation

As we mentioned earlier, a performance test may bring numerous benefits for organizations that apply this type of test, since they can predict the scalability of their applications. It provides better control of the application, also avoiding financial losses because of possible application unavailability due to excessive requests (SUBRAYA, 2006).

Meier *et al.* (MEIER et al., 2007) list some of the main reasons for conducting performance testing, for example: assessing release readiness, assessing infrastructure adequacy, assessing the adequacy of developed software performance, and improving the efficiency of performance tuning.

Some papers offer a basis for those who are interested in performance testing to guide themselves during its application, such as the works of Subraya (SUBRAYA, 2006) and the certification material offered by the International Software Testing Qualifications Board (ISTQB) (BLACK; ROMMENS; AALST, 2017). In addition, there are several materials available on different websites and blogs spread around the Internet, due to the constant demand for information related to performance testing. Thus, we believe that a repository that gathers this information may facilitate applying performance testing in different organizations' particularities and workflows. Therefore, an information repository as a framework or even a knowledgeable guide, similar to those found in other areas such as Project Management Body of Knowledge (PMBOK) and Software Engineering Body of Knowledge (SWEBOK), may bring numerous benefits to those who are interested in performance testing, whether they are from academia or the software industry.

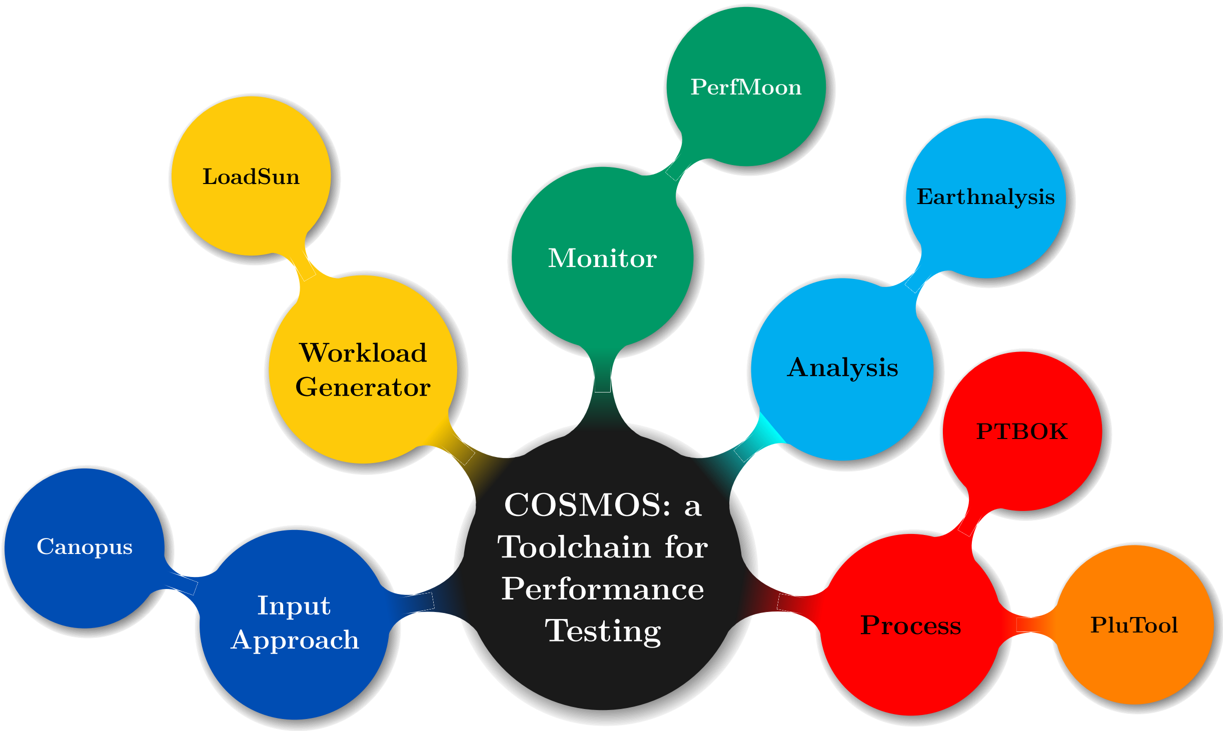
The Performance Testing Body of Knowledge (PTBOK) comes from one of the research lines of the research group of the Laboratory of Empirical Studies in Software Engineering (LESSE)[1] being part of a solution called COSMOS. With COSMOS, we seek to develop a performance testing solution that supports modeling activities, generating synthetic workloads, monitoring, and analyzing results, to support the performance testing process in Web applications. The PTBOK has the role of providing performance testing guidelines to COSMOS tools.

Figure 1 shows COSMOS composition details, through a representation of its architecture. Each node represents specific solutions that contribute to the general COSMOS solution. Canopus is an input approach, a domain-specific language for performance testing. PTBOK provides a process to guide the conduct of the test. The other nodes present tools for analysis (EarthAnalysis), monitoring (PerfMoon), workload generation (LoadSun), and a solution for the integration of the process with the tools (PluTool). PTBOK is directly related to the solutions for Analysis, Monitor, and Workload Generator, since the concepts mapped in the process provide the basis for these projects. In addition, there is also a direct relationship with Plug Tool, which will be responsible for orchestrating and integrating the process with the tools that are under development/evolution.

There are some solutions adopted for modeling software processes. Business Process Model and Notation (BPMN) (OMG; PARIDA; MAHAPATRA, 2011), Software & Systems Process Engineering Metamodel (SPEM) (OMG, 2006), and Essence (SUBMITTERS, 2012) are some of the best known and most adopted solutions for this type of modeling. To model PTBOK, we chose SPEM because it provides a suitable metamodel for modeling software processes and has tooling support that allows the publication of the modeled process on a web page, keeping the artifacts and components of the process

[1] <www.lesse.com.br>

Figure 1 – `COSMOS` architecture.



Source – Author.

organized in a repository as a knowledge management system.

After PTBOK modeling, we evaluated it with performance testing experts, from an industry perspective. Then, we conducted a survey that allowed us to understand the positive points, applicability, and improvements to PTBOK. In addition, we initially conceived a tool that will support the PTBOK activities from a scientific initiation project.

1.2 Objectives

Our main goal is to design the PTBOK. For this reason, we divided the main goal into the following specific goals:

1) to promote performance testing state of the art and state of practice;
2) to characterize the domain of performance testing;
3) to point out a fundamental knowledge for establishing disciplines of undergraduate and graduate curricula;
4) to provide core knowledge for certification professional providers.

1.3 Main Contributions

The main contributions of our research are as follows.

1) An SLR collecting, analyzing, and discussing thirty seven (37) different works on performance testing area;
2) A Feature Model with the main concepts related to performance testing;
3) A Survey with relevant information to the performance testing process from an industry perspective;
4) Modeling of PTBOK in a concise metamodel.

1.4 Organization

We organized this document as follows.

- Chapter 2 details our research design and the schedule;
- Chapter 3 describes the theoretical grounds of our work, addressing the main concepts related to performance testing, body of knowledge, and process modeling;
- Chapter 4 presents an SLR performed for collecting performance testing data from the literature, which we used for PTBOK definition;
- Chapter 5 presents a Survey that we performed for collecting performance testing data from the industry, which we used for PTBOK definition;
- Chapter 6 explains and details the PTBOK;
- Chapter 7 discusses PTBOK assessment that we conducted through a survey;
- Chapter 8 presents the conclusion and also our future perspectives regarding this study.

2 METHODOLOGY

In this chapter, we describe the methodology used to support our work. Section 2.1 introduces what it is and why it is important. In Section 2.2 the research is classified according to Prodanov e Freitas (2013), and the research design is shown in Section 2.3.

2.1 Introduction

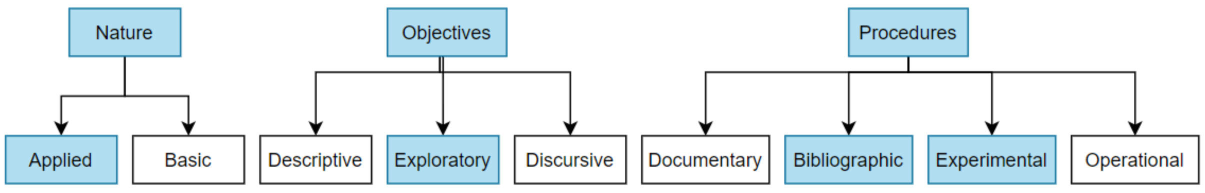
Research is the basic activity of Science in its inquiry and construction of reality (LAUDAN, 1981). Through research, we may acquire knowledge about the reality of the world and contribute theoretically, promoting thought and action. Moreover, by research we seek to clarify issues through the scientific method, starting from a problem that does not yet have satisfactory knowledge.

The application of the scientific method must have an emphasis on the applied methodology and not only on the obtained results. The factors that influence the classification criteria are the approach, objectives, situations, fields, and objects of the provided study.

2.2 Research Classification

We classify our search according to Prodanov e Freitas (2013) classification scheme. Figure 2 shows the classification according to its nature, objectives, and procedures, the highlighted terms classify this research and are described in this section.

Figure 2 – Research Classification



Source – Adapted from Prodanov e Freitas (2013)

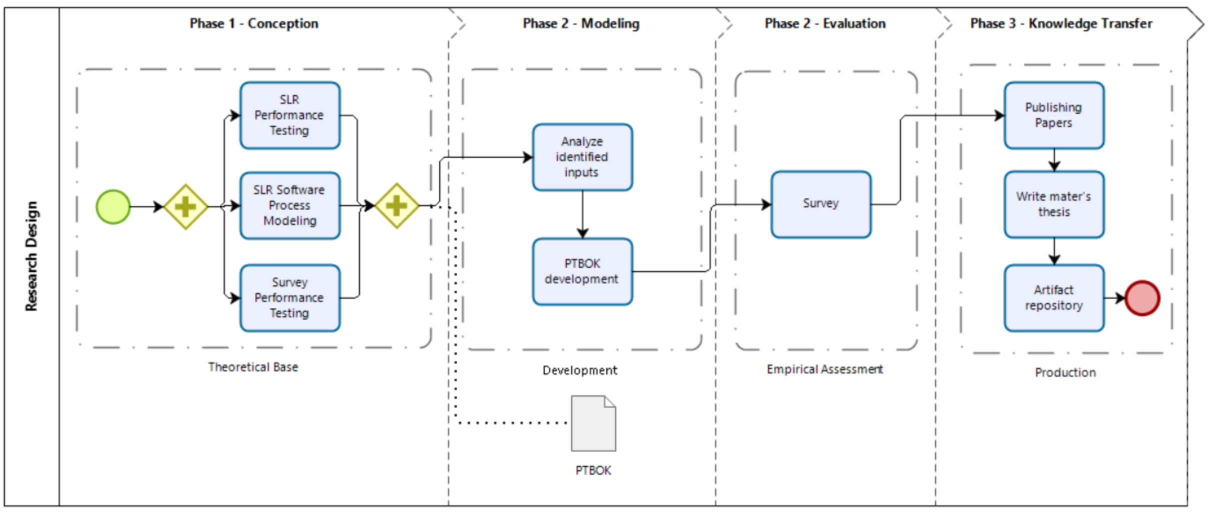
From the nature point of view, this is **Applied Research**, which aims to generate knowledge for practical application directed to the solution of specific problems.

This research is classified as **Exploratory Research** since we seek to acquire information that allows us to better understand a certain subject for a possible transfer of knowledge. Concerning procedures, our research is classified as **Bibliographic** and **Experimental Research**, because of the SLR and Survey that we carried out.

2.3 Research Design

In Figure 3, we show the research design that we followed. We divided this design into three (3) main phases. In the first phase, conception, we mapped through two empirical studies the foundation for PTBOK. Then, in the evaluation phase, we intend to evaluate the modeled process. Thus, we carried out a survey. Finally, phase 3 presents ways that we intend to act to carry out knowledge transfer, employing paper's publication, this master's thesis, and an artifact repository, where we will make our work outputs available.

Figure 3 – Research Design



Source – Author.

2.4 Chapter Summary

In this chapter, we presented our research methodology. Through this, it is possible to have a high-level view of all of our research, since it contemplates the main stages planned for carrying out the work.

3 BACKGROUND

In this chapter, we present the background of our research, which enables some understanding of the main concepts related to our work. In Section 3.1, we show an overview of performance testing, followed by an architectural example of this testing type. In Section 3.2, we introduce a brief explanation of a body of knowledge, as we seek to instantiate a body of knowledge for an area. Then, Section 3.3 offers process modeling definition, more specifically about Software & Systems Process Engineering Metamodel (SPEM), which we use for modeling PTBOK. Finally, we show the lessons from this chapter in Section 3.4.

3.1 Performance Testing

Performance testing brings the possibility to plan, execute, monitor, and analyze the results of a system under certain conditions, thereby obtaining the possible expected behavior of a given software, when subjected to these conditions (BERNARDINO; ZORZO; RODRIGUES, 2016). According to Freitas e Vieira (2014), performance testing is a test that aims to evaluate the performance of the system at a given load scenario. In summary, performance testing provides a load simulation and measurement to detect bottlenecks and the breaking point in which a system crashes under a certain workload.

Woodside, Franks e Petriu (2007) define Software Performance Engineering (SPE) as representing the entire collection of software engineering activities and also related analyses throughout the software development cycle, which are set to meet 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 focuses on the early cycle and the latter one in the late cycle of the software development process. Hence, this performance testing is associated with a measurement-based approach.

According to Meier et al. (2007), 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, submit an application to a workload. It should be noticed that the load test is conducted to assess if the given system meets specified non-functional requirements. A stress testing, on the other hand, checks a system under normal conditions operation, as well as under normal workload. Through stress testing, it is also possible to know the behavior of the SUT when submitted to heavy workloads. Moreover, Molyneaux (2009) includes soak, or scalability, testing, in a way that soak testing may subject the SUT to a load for a long period, in which some problems dismissed in other categories may become noticeable.

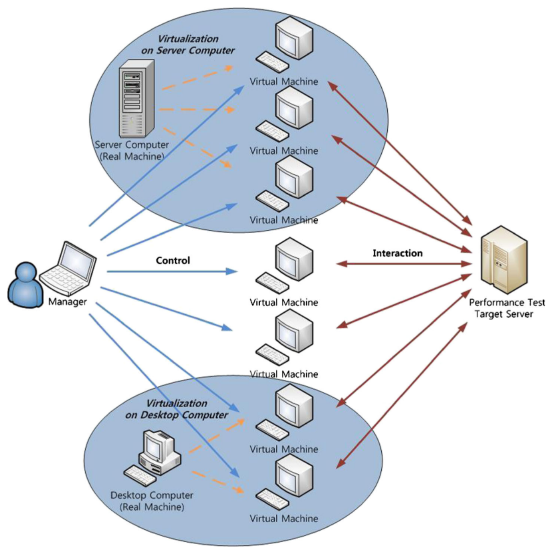
Workload generation is critical to the performance testing process. It is not possible to test the system without some load type, whether simulated or not, being applied to the system. Then, we address that the performance metrics are critically dependent on the workloads processed by the system under test (FERRARI, 1984).

3.1.1 System Architecture for Performance Testing

Kim, Kim e Chung (2015) propose a system architecture for performance testing based on virtualization. In Figure 4 the performance testing target is represented for PTTS (Performance Test Target Server), which will perform the necessary interactions with the Virtual Users (VU) generated by the Virtual Machines (VM). It has a server application and responses to the request of a client application. VM generates a load on PTTS, so it installs the client application and executes the appropriate load for each test scenario.

This architecture model may reduce computer resource consumption for performance testing. Since virtualization may help to reduce the number of physical computers for performance testing, it can also generate several VMs on physical computers (KIM; KIM; CHUNG, 2015).

Figure 4 – System architecture (KIM; KIM; CHUNG, 2015).


Virtualization on Server Computer
Server Computer (Real Machine)
Virtual Machine
Virtual Machine
Virtual Machine
Manager
Control
Virtual Machine
Interaction
Performance Test Target Server
Virtual Machine
Virtualization on Desktop Computer
Desktop Computer (Real Machine)
Virtual Machine
Virtual Machine

Source – (KIM; KIM; CHUNG, 2015).

3.2 Body of Knowledge

A Body Of Knowledge (BOK) is defined by a set of teaching, skills, and abilities necessary to carry out activities in a field. In addition, a BOK must bring together not only desirable items or characteristics for the execution of the tasks, but also gather in a self-contained manner what is necessary so that the specified domain is understood by all interested parties (BOURQUE et al., 1999).

Another characteristic of a BOK is the constant evolution and updating since the mapped knowledge must come from both the state-of-the-art and the state-of-practice, that is, as well as techniques, methods, and concepts, tools of the field evolved, this evolution must be present in the BOK content (ABRAN et al., 2004).

Some well-known BOKs served as a basis for understanding the structure and organization that we could use to create PTBOK. Among these, we highlight the Software Engineering Body of Knowledge (SWEBOK) (BOURQUE et al., 1999), which brings together the main concepts of the software engineering area and also Project Management Body of Knowledge (PMBOK) (PMI, 2000), this one with a focus on project management. There is also the Business Analysis Body of Knowledge (BABOK)(A. . . ,) that encompasses the concepts in the area of business analysis.

3.3 Software & Systems Process Engineering Metamodel (SPEM)

Software and Systems Process Engineering Metamodel (SPEM) is a process engineering metamodel and a conceptual framework that enables provide the concepts needed to model, document, present, manage, exchange, and perform development methods and processes. The implementation of this metamodel is directed to process engineers, project leaders, project managers, and developers who are responsible for maintaining and implementing individual processes or for their organizations (OMG, 2006).

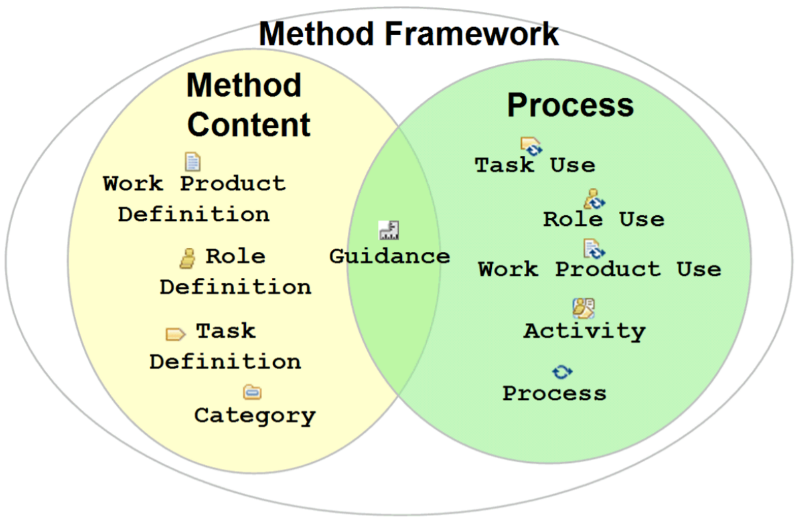
SPEM 2.0 is used to define software development processes and systems as well as their components. Its main aim is to support a wide range of development methods and processes of different styles, cultural backgrounds, levels of formalism, life cycle models, and communities. This metamodel enables the developer to choose the generic approach to behavioral modeling that best meets his needs. It even provides a specific framework for enhancing these generic behavioral models, which are characteristic for describing different processes, focusing on providing the additional information structures processes modeled with Unified Modeling Language (UML) 2.0 or Business Process Model and Notation (BPMN)/ Business Process Definition Metamodel (BPDM) activities, with the purpose to describe a real process.

One of the main features of the SPEM is the possibility of the specification of different processes from a common knowledge base regardless of the specific process. A clear separation is defined between method content and process content, the content represents a knowledge base while the process represents a process is specified. That is the knowledge base that stores all the information required to process the life cycle.

Figure 5 shows the SPEM structure, which consists of two main parts, Method Content, and Process. The first one is composed of Work Product Definition, Role Definition, Task Definition, and Category. Method Content provides the concepts for SPEM 2.0 users and organizations to build up a development knowledge base that is independent of any specific processes and development projects. The process is composed of Task

Use, Role Use, Work Product Use, Activity, and Process and follows method content specification to instantiate a process.

Figure 5 – SPEM structure (ELVESAETER; BENGURIA; ILIEVA, 2013)



Source – (OMG, 2006)

3.4 Chapter Summary

In this chapter, we presented our theoretical foundation. We detailed the main concepts of Performance Testing. Also, we have included some definitions about Body of Knowledge and also about the SPEM metamodel, which we use for modeling PTBOK.

4 SYSTEMATIC LITERATURE REVIEW

In this chapter, we present the SLR that we carried out. Section 4.1 details the protocol we followed for conducting the study. In Section 4.2, we discuss the results that we obtained, which explores the selected works so that we may answer our research questions.

4.1 Protocol

SLR scope is Performance Testing study area, seeking out 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 (2007) in SLR. Also, we used the Thoth tool to support the conduct of the study (MARCHEZAN et al., 2019). The GQM (Goal, Question, Metric) paradigm (KOZIOLEK, 2008) usage means to resume the review scope, which may be observed in Table 1.

Table 1 – Objective according to the GQM paradigm (KOZIOLEK, 2008).

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*

Source – Author.

4.1.1 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 activities or task flows performed in performance testing?

4.1.2 Question Structure

Research Questions (RQs) are designed using (PICOC) (WOHLIN et al., 2012) criteria, that take into consideration the Population, Intervention, Comparison, Outcome, and Context.

Population: published research on software;

Intervention: performance testing;

Comparison: general comparison of the retrieved processes;

Outcome: published papers on Performance Testing;

Context: software testing practice and research.

4.1.3 Search Strategy

To perform the proposed search, we selected the following databases:

- Scopus[1]
- IEEE Xplore Digital Library[2]
- ScienceDirect[3]
- Engineering Village[4]
- ACM Digital Library[5]

These databases were chosen because they store the main publications in the computer science field and they also offer 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 2.

Selected Databases: Scopus, IEEE, Science Direct, Engineering Village, ACM.

Table 2 – 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)`

Source – Author.

4.1.4 Selection Criteria

Inclusion Criteria (IC) and Exclusion Criteria (EC) were defined and applied to filter in the initial search. An inclusion criterion is a feature that implies including a given study in the scope of the current research. An exclusion criterion, by its turn, based on some reason excluded a given study from the body of the research. Besides, for a study

[1] Scopus:<https://www.scopus.com>
[2] IEEE: <https://ieeexplore.ieee.org/>
[3] ScienceDirect:<https://www.sciencedirect.com>
[4] Engineering Village:<https://www.engineeringvillage.com>
[5] ACM DL:<https://dl.acm.org/>

inclusion, the paper must satisfy at least one inclusion criterion and at least one exclusion criterion as a way to exclude the study from our analysis.

Therefore, the inclusion and exclusion criteria we defined were as follows:

- **IC1.** The study addresses web performance testing;
- **IC2.** The study proposes some method, process, framework, approach, toward or guideline related to performance testing;
- **IC3.** The study describes or presents some activity, step, stage, phase about any performance testing process;
- **EC1.** The study is less than 5 pages;
- **EC2.** The study is in a language other than English;
- **EC3.** The study is not available for download;
- **EC4.** The study is not related to performance testing in the software area;
- **EC5.** The study is not a primary study;
- **EC6.** The study has received note 1.0 or less in the quality assessment.

4.1.5 Quality Assessment Criteria

To evaluate selected studies' relevance and also answering some research questions, we used the Quality Assessment (QA) criteria. The quality assessment criteria are featured and may be exploited in two stages: the first stage as being an individual evaluation of each researcher, to reduce bias probability; the second stage is when researchers should reach a consensual note about publications in a "divergent state" in a quality measurement grade.

Each of the cited QA criteria is evaluated by each researcher, according to the following degree: Yes (Y) = 1; Partial (P) = 0.5; No (N) = 0. So the total score ranging the five questions can result in: 0-1.0 (poor); 1.5 or 2.0 (regular); 2.5 or 3.0 (good); 3.5 or 4.0 (very good); and 4.5 or 5.0 (excellent). Each of the criteria and their possible evaluations are described as following:

QA1: Did the study describe the performance testing profiles/roles, methods, artifacts, or approaches?

Y: The study describes completely a performance testing profile/role, method, artifact, or approach;

P: The study describes partially a performance testing profile/role, method, artifact, or approach;

N: The paper does not describe a performance testing profile/role, method, artifact, or approach.

QA2: Did the study mention the performance testing stages or phases?

Y: The study mentions completely a performance testing stage or phase;

P: The study mentions partially a performance testing stage or phase;

N: The study doesn't mention a performance testing stage or phase.

QA3: Did the study explain the performance testing activities, steps, or tasks?

Y: The study explains completely a performance testing activity, step, or task;

P: The study explains partially a performance testing activity, step, or task;

N: The study does not explain a performance testing activity, step, or task.

QA4: Did the study report activities/tasks flow performed by performance testing?

Y: The paper reports completely a performance testing activities/tasks flow;

P: The paper reports partially a performance testing activities/tasks flow;

N: The paper does not report performance testing activities/tasks flow.

QA5: Did the study present an evaluation for its proposal?

Y: The paper presents completely the evaluation;

P: The paper presents partially the evaluation;

N: No evaluation was presented.

4.1.6 Data Extraction Strategy

To extract the relevant data from the selected studies, a specific form was produced. The following data were extracted for each study:

DE1. Title;

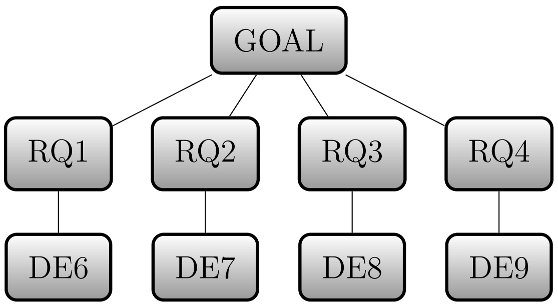
DE2. Author;

DE3. Year;

DE4. Conference;

DE5. Addresses;

Figure 6 – GQM Structure



Source – Author.

DE6. Performance Testing Profiles, Roles, Artifacts, Methods, or Approaches;

DE7. Performance Testing Stages or Phases;

DE8. Performance Testing Activities, Steps, or Tasks;

DE9. Performance Testing Activities/Task Flows;

DE10. Empirical Evaluation.

To support our extraction data election, we present, in Figure 6 the relationship between research questions and derived data extraction through the GQM paradigm (KOZIOLEK, 2008).

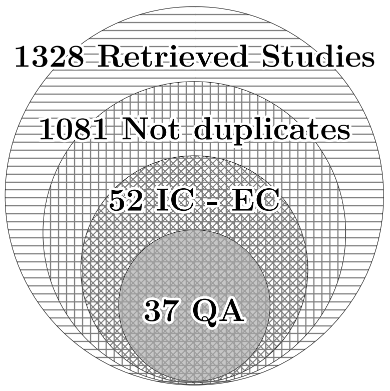
4.1.7 Selection Process

1. **Pilot Search Strategy:** To verify the quality of the proposed search string, the approach called Search-Based String Generation (SBSG), proposed by Souza (SOUZA et al., 2018) was applied. The approach is based on precision and sensitivity indexes calculation. Precision is the ability to identify the number of *irrelevant* studies, while sensibility is a way 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; 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 based on 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;

Figure 7 – Onion diagram showing the number of papers after each step of the SLR process.



Source – Author.

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 1328 results (see Figure 7). 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 (see Table 3) reduced the number of results to thirty seven (37) papers.

4.2 Results and Discussion

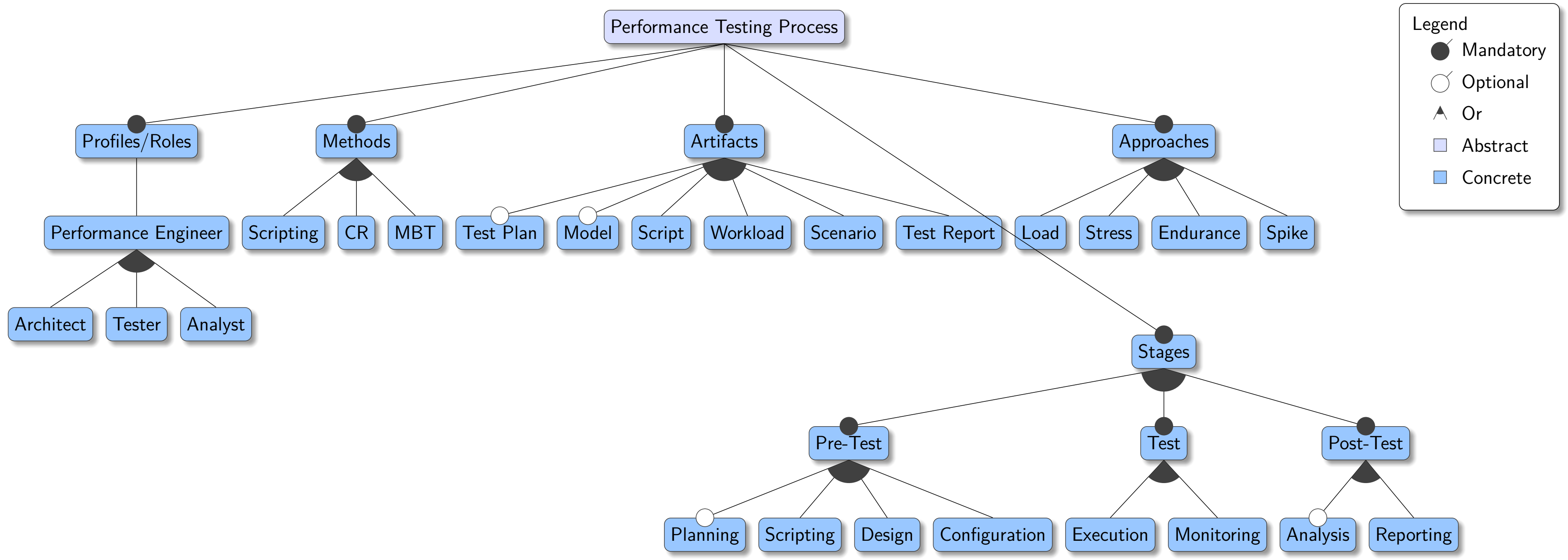
In this section, we present the SLR results, in which thirty seven (37) studies are discussed to respond to defined research questions. Figure 8 provides us an overview of the results by a feature model. Nodes Test Plan, Model, Planning, and Analysis are optional. For instance, in an approach that does not use a model as an artifact, this is not required.

Table 3 – Quality Assessment Results

Studies Reference	**Year**	**QA1**	**Quality Assessment QA2**	**QA3**	**QA4**	**QA5**	**Evaluation Score**
Tselikis, Mitropoulos e Douligeris (2007)	2007	T	P	T	P	N	3.0
Sharifi, Tasharrofi e Mahmoudzadeh (2005)	2005	T	T	P	T	N	3.5
Hadharan et al. (2000)	2000	N	T	T	N	N	2.0
Anderson et al. (2006)	2006	T	P	T	N	N	2.5
Pfau, Smeddinck e Malaka (2017)	2017	N	P	P	P	N	1.5
Sprenkle et al. (2005)	2005	T	T	P	N	T	3.5
Yin et al. (2008)	2008	P	T	T	P	T	4.0
Huang et al. (2011)	2011	P	T	T	T	N	3.5
Rodrigues et al. (2014)	2014	P	P	T	T	T	4.0
Braga et al. (2018)	2018	T	P	T	P	T	4.0
Liu et al. (2018)	2018	T	P	T	P	P	3.5
Xia et al. (2006)	2006	N	T	P	P	P	2.5
Hanmer e Letourneau (2003)	2003	T	P	T	P	N	3.0
Arora (2016)	2016	P	P	P	N	N	1.5
Mirshokraie, Mesbah e Pattabiraman (2015)	2015	N	T	P	N	P	2.0
Gao e Li (2011)	2011	N	P	T	P	P	2.5
Kun et al. (2008)	2008	N	P	T	P	P	2.5
Marszalkowski (2012)	2012	N	P	P	P	N	1.5
Boone et al. (2010)	2010	T	P	T	P	P	3.5
Garg, Singla e Jangra (2016)	2016	N	P	T	N	N	1.5
Ster et al. (2011)	2011	T	P	T	P	P	3.5
Camargo et al. (2016)	2016	N	P	T	N	P	2.0
Chen et al. (2010)	2010	P	P	P	N	N	1.5
Juric et al. (2006)	2006	N	P	P	N	P	1.5
Xu et al. (2014)	2014	T	P	T	P	T	4.0
Pons (2005)	2005	N	P	T	N	P	2.0
Souza e Travassos (2017)	2017	N	P	T	P	T	3.0
Subraya (2006)	2006	T	T	T	T	N	4.0
Xia et al. (2010)	2010	T	P	T	P	P	3.5
Snodgrass (1988)	1988	N	P	T	P	N	2.0
Bernardino, Zorzo e Rodrigues (2016)	2016	T	T	T	T	T	5.0
Putri, Hadi e Ramdani (2017)	2018	T	P	T	P	T	4.0
Ali e Badr (2015)	2015	P	P	T	P	N	2.5
Gias et al. (2013)	2013	T	P	T	P	N	3.0
Freitas e Vieira (2014)	2014	T	P	T	P	N	3.0
Rodrigues et al. (2015)	2015	T	T	T	T	N	4.0
Meier et al. (2007)	2007	T	P	T	T	N	3.5

Source – Author.

Figure 8 – Taxonomy of performance testing tools represented by feature model.



Source – Author.

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

In this RQ, we explain the profiles/roles, methods, artifacts, or approaches identified in the selected studies.

- **Profiles/Roles**

  Profiles/Roles related to Performance Testing are as follows:

  - **Performance Engineer:** The performance engineer must be able 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 refer directly or indirectly to this role (SUBRAYA, 2006) (XU et al., 2014) (STER et al., 2011);
  - **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;
  - **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 measuring performance. A few papers bring this role within another nomenclature as User and Developer (TSELIKIS; MITROPOULOS; DOULIGERIS, 2007) (PFAU; SMEDDINCK; MALAKA, 2017). We merged these terms in Performance Tester, once we believe that is more suitable for this context;
  - **Performance Analyst:** The performance Analyst has participated in early and late performance testing phases. This role is responsible for initial testing planning and documentation, providing input to subsequent phases, design, and configuration; This role is also present after testing execution, hence, it is employed in the analysis and reporting phases. This role is not directly reported in the selected papers. However, Subraya (SUBRAYA, 2006) refers to their activities, without specific nomenclature.

After analyzing selected studies, we identified the following four (4) profiles/roles: It is noteworthy that the roles above listed are technical, not covering management levels of organizations.

- **Artifacts**

  Some artifacts are presented to support performance testing activities. The most relevant artifacts in this context are:

  - **Performance Test Plan:** A 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 focus on this phase (MEIER et al., 2007) (FREITAS; VIEIRA, 2014) (YIN et al., 2008) (HUANG et al., 2011);
  - **Model:** This artifact is used as input in a 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:** A 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 automatically or manually. In the former, scripts are generated through tools that use capture and replay mechanisms (SUBRAYA, 2006). On the latter one, in manual form scripts, they are generated through a programming language code;
  - **Workload:** This artifact is responsible for modifying the SUT situation through its different configurations. A workload may vary based on the test approach and it includes the 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; SMED-DINCK; MALAKA, 2017);
  - **Performance Scenario:** Meier et al. (2007) define a scenario as a set of steps in an application. Moreover, a scenario may map a given application context, within a determinate workload for a user profile, it should be modeled based on usage patterns and log files. In other words, a scenario must reflect real or expected system usage for performance testing;
  - **Performance Test Report:** Test execution should produce data for reporting. A technical report must contain test results, organized in a way that allows their interpretation by stakeholders. Meier et al. (2007) list 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 references section.

- **Methods**

  Three methods are related to performance testing conducting (RODRIGUES et al., 2015):

  - **Scripting:** This method involves technical support by the manual script where the performance tester writes a set of code statements, which are inputs 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 model predictions. It has some advantages such as enabling the application of models for appropriate testing models creation, as well as its use in performance testing;
  - **Capture and Replay** This method consists 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.

- **Approaches**

  Subraya (2006) presents a set of four (4) performance testing approaches called LESS (Load, Endurance, Stress, Spike). These approaches are discussed below:

  - **Load Testing:** A load is the number of users that compete to increase the traffic of the application. It is useful for determining the breaking point and checking when bottlenecks begin to emerge;
  - **Endurance Testing:** 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 perform on a normal load or a stress load, but the main focus of this approach is the test duration;
  - **Stress Testing:** Stress testing is similar to load testing. However, stress testing aims to check how the application handles its limit. Therefore, it helps to identify the load that the system can handle before breaking down or degrading quickly;
  - **Spike Testing:** A spike testing is conducted to verify application behavior under a surge in a short duration. The application is submitted to a sudden load increase.

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

For our purpose, stages were mapped as being the activities group at a high level, which may have one or more phases.

We identified three (3) stages and eight (8) phases in the performance testing context. The stages and phases are as follows:

- **Pre-Test**: This stage comprises previous phases to test execution. The test definition and preparation occur in this stage. The Pre-Test stage has four (4) phases:
  - **Planning:** In this phase, it 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 relevant to the test;
  - **Scripting:** This phase involves activities that focus on script elaboration, which can be obtained by different means. For instance, supported by models and by using the MBT or CR for an automatic generation or also employing coding, where scripts are made from a specific programming language;
  - **Design:** Using the test specifications defined in the planning phase, performance testing is designed taking into account environments particularities and performance testing goals;
  - **Configuration:** It is the last phase before test execution. In this state, adjustments and setting performance testing are made. Issues like workload type, performance testing type and tool functionalities should be considered as well as infrastructure issues.

- **Test**: The Test stage is related to the execution of the test and it is performed after the pre-test stage.
  - **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, the test should provide mechanisms to collect necessary metrics, which were defined in the Pre-Test stage.
  - **Monitoring:** Defined metrics as throughput, response time, and 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 the post-test stage.

- **Post-Test**: This stage encompasses the phases of Analysis, and Reporting.
  - **Analysis:** This phase aims to carry out the analysis of the test results, 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 to the analysis by the performance engineer in a decision making process;

Table 4 – Performance Testing Stages/Phases and Evaluation type.

	Pre-Test				Test		Post-Test		Evaluation	
Ref.	**Planning**	**Scripting**	**Design**	**Configuration**	**Execution**	**Monitoring**	**Analysis**	**Reporting**	**Case Study**	**Experiment**
Tselikis, Mitropoulos e Douligeris (2007)	✓				✓	✓	✓			
Sharifi, Tasharrofi e Mahmoudzadeh (2005)		✓			✓					
Hadharan et al. (2000)					✓					
Anderson et al. (2006)					✓					
Pfau, Smeddinck e Malaka (2017)					✓	✓	✓			
Sprenkle et al. (2005)					✓					✓
Yin et al. (2008)	✓		✓		✓				✓	
Huang et al. (2011)	✓	✓			✓	✓				
Rodrigues et al. (2014)	✓	✓			✓	✓				✓
Braga et al. (2018)	✓				✓	✓				✓
Liu et al. (2018)					✓					✓
Xia et al. (2006)		✓								✓
Hanmer e Letourneau (2003)	✓	✓			✓	✓				
Arora (2016)	✓	✓			✓	✓				
Mirshokraie, Mesbah e Pattabiraman (2015)					✓	✓				
Gao e Li (2011)		✓			✓	✓	✓	✓		
Kun et al. (2008)					✓	✓			✓	
Marszalkowski (2012)					✓	✓				
Boone et al. (2010)					✓					✓
Garg, Singla e Jangra (2016)					✓					
Ster et al. (2011)					✓	✓	✓		✓	✓
Camargo et al. (2016)					✓	✓	✓	✓	✓	
Chen et al. (2010)		✓								
Juric et al. (2006)					✓					
Xu et al. (2014)	✓	✓		✓	✓	✓	✓	✓	✓	
Pons (2005)		✓						✓	✓	
Souza e Travassos (2017)	✓	✓			✓	✓		✓	✓	
Subraya (2006)	✓	✓	✓	✓	✓	✓	✓	✓		
Xia et al. (2010)					✓	✓		✓	✓	
Snodgrass (1988)					✓	✓				
Bernardino, Zorzo e Rodrigues (2016)		✓			✓	✓	✓	✓	✓	
Putri, Hadi e Ramdani (2017)	✓				✓	✓		✓	✓	
Ali e Badr (2015)	✓				✓	✓				
Gias et al. (2013)	✓				✓	✓				
Freitas e Vieira (2014)	✓				✓					
Rodrigues et al. (2015)		✓			✓	✓				
Meier et al. (2007)	✓	✓	✓	✓	✓	✓	✓	✓		

Source – Author.

- **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 interprets performance testing report results.

In Table 4, the relation between selected paper, stage, and phase are related.' It is possible to verify that most of the studies address test execution. Another 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 of this type were found, followed by experiments with seven (7) studies that used this approach to assess the study. Another relevant question is that the majority of studies are not focused on all phases and stages of performance testing, because they focus on some specific phases.

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

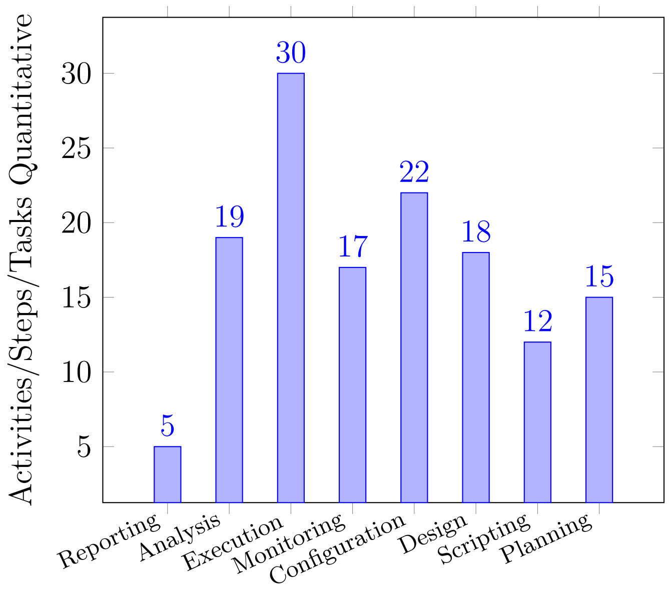
Based on the selected papers, we found one hundred and thirty eight (138) performance testing activities, steps, or tasks. In Figure 9 it is possible to identify that assumption, as well as the trend evidenced in previous research questions. There is a greater concentration of activities in the test execution phase, where we identified forty seven (30) activities related to this phase. Some performance testing tools work with monitoring and execution independently. However, others do not distinguish between these phases. The other phases have a similar commensurate of activities, ranging from fifteen (15) to twenty two (22) activities, except the 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 the 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.

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

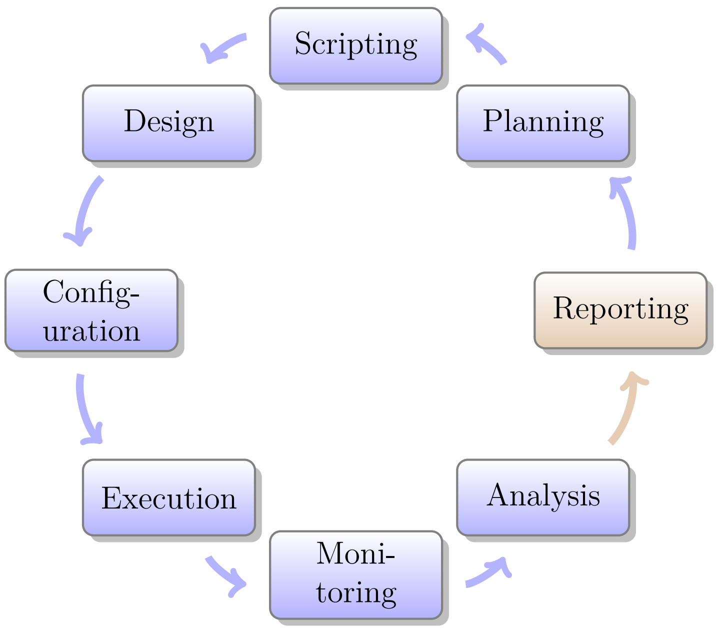
The mapping of the stages, phases and subsequent activities reported in the selected studies allowed us to organize the phases as shown in Figure 10. This flow circularly presents the phases. Performance testing may be thought of as a sequential activity and may be instantiated as many times as necessary. In this flow, the sequence starts in the Planning, following the seven (7) next phases until completing the cycle with the Reporting phase, which is highlighted to mark to the end. Another reason that motivated

Figure 9 – Relation Activities/Steps/Tasks Quantitative vs Phases.



Source – Author.

Figure 10 – Performance Testing Activities/Tasks Flow



Source – Author.

us to model the flow in this manner 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, independently on the activity described to contemplate the phases in their totality or not.

4.3 Chapter Summary

In this chapter, we presented 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 and by a feature model that encompasses the whole SLR results. The paper "Systematic Literature Review on Web Performance Testing", which presents the results of this review, was accepted at the *IV Regional School of Software Engineering*[6].

[6] <https://sol.sbc.org.br/index.php/eres/article/view/13739/13587>

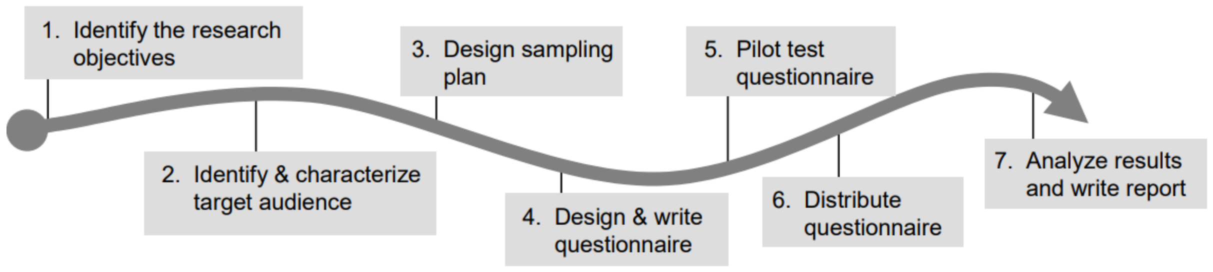
5 SURVEY

In this chapter, we present the survey that was carried out. This study had as the main objective, to gather information about the performance testing from a professional perspective. Section 5.1 details the survey protocol. In Section 5.2, we present the main threats to the study and some ways to try to mitigate them. Section 5.3 provides a discussion about results that we found from survey conduction.

5.1 Survey Protocol

Our survey protocol is based on Kasunic (2005), which defined a seven-stage, end-to-end process for survey conducting. Figure 11 provides us an overview of this process.

Figure 11 – Survey flow by Kasunic (KASUNIC, 2005)


1. Identify the research objectives
2. Identify & characterize target audience
3. Design sampling plan
4. Design & write questionnaire
5. Pilot test questionnaire
6. Distribute questionnaire
7. Analyze results and write report

Source – Author.

5.1.1 Identify the research objectives

This stage aims to elucidate what we want to accomplish through this survey. Therefore, we aim to identify major inputs related to Web performance testing, under a technical view, gathering opinions from the industry. We hope that from the data obtained through this survey, it will be possible to gather insights that support us in the performance testing body of knowledge creation, which must contemplate the state of practice of the area under review. In other words, we look for methods, guidelines, artifacts, activities, steps, tasks, and techniques related to performance testing.

5.1.2 Identify and characterize the target audience

In this stage, we expect to discover who, specifically, will respond to the survey. Thus, our target audience may encompass industry professionals who work directly or indirectly with performance testing that characterize the relevant population for our study. These subjects may have different roles in their companies, the main relevance for being part of the research is the knowledge about the performance testing process in their organizations.

5.1.3 Design the sampling plan

Design the sampling plan is related to some questions:

- How large is the target audience population?
- Can the target audience be enumerated?
- How will you ensure that those who respond to the survey are representative of the target audience?

Based on the questions listed above and to obtain a relevant sample for the research, we adopted two (2) different strategies. The first one is through direct contact with IT companies as well as specific performance testing organizations. The second one is the survey dissemination in the main social networks and discussion forums. To increase the likelihood of adherence to the questionnaire, we adopted some criteria for the disclosure of the questionnaire:

- The discussion group must have at least 1000 members[1];
- The discussion group should be from a performance testing-related subject.

We chose the following social networks for the survey: Facebook[2] (11 groups), LinkedIn[3] (5 groups), Stack Overflow[4]. In addition, the questionnaire was also disclosed in a WhatsApp[5] discussion group about performance testing with 253 active members[6].

5.1.4 Design and write the questionnaire

After the early stages, the survey objectives must be worded into the questionnaire. Hence, we followed some best practices presented in Shull, Singer e Sjøberg (2007) study:

- Present open and closed questions;
- Avoid yes/no questions;
- Provide extra space for comments;
- Use pattern tools;
- Easier questions should be asked first;
- Explain how confidentiality will be preserved;

[1] We adopted this criterion so that we had a greater probability of responses due to a large number of groups
[2] Facebook:<https://www.facebook.com>
[3] LinkedIn:<https://www.linkedin.com>
[4] Stack Overflow:<https://www.stackoverflow.com>
[5] WhatsApp:<https://www.whatsapp.com>
[6] Groups details in<http://bit.ly/2Q3Picz>

- Present research credentials, university name, researcher's name, *etc.*

As a tool for creating and applying the questionnaire, we chose the LimeSurvey[7], which is an open-source solution that offers appropriate functionality for our research.

We intend to get data about performance testing experience from subjects, therefore this survey is classified as cross-sectional. Moreover, we applied a self-administered questionnaire, where the participants are responsible for reading and answering the questions.

Firstly, it was presented the Informed Consent Form (ICF), followed by profile questions. Then, specific performance testing questions were divided into five (5) stages. Our focus was to separate the content seeking to facilitate the fill and understanding of the questionnaire.

Profile questions were elaborated aiming to verify the knowledge level from subjects, seeking to level participants and to divide them into groups for future results analysis.

The elaborated performance testing stages questions focus on main inputs related to performance testing. These inputs are composed of roles, profiles, activities, phases, tasks, steps, and the performance testing general flow.

- **Profile Questions**

  Profile questions are as follow:

1) What is the highest level of education you have completed?

2) How many years of experience do you have in the performance testing area in the industry context?

3) How many years of experience do you have in the performance testing area in the academic context?

4) Which one of the following best describes your organization?

5) What is your main job position?

6) Finally, please indicate your name, your organization's name, and your email address below if you would like to contribute to our research or to receive a summary of the results. Your data will be combined with the data of other respondents and shared only in aggregate:

   (Name, Name of the organization, and e-mail address were required in a non-mandatory way).

[7] LimeSurvey:<https://www.limesurvey.org/>

- **Performance Testing Questions**

  Performance testing questions are the following:

1) The maturity level of performance testing activities[8].
2) Does your organization follow a defined performance testing methodology/process, or is it done *ad hoc*?
3) What are the roles, functions (positions in the company) involved in performance testing?
4) What are the main activities performed by the performance testing team on software projects of your organization?
5) Is there a specific performance testing infrastructure in your organization? If so, is it shared or dedicated? If so, is it physically, virtualized, or in the cloud?
6) How is the process of setting performance testing scenarios and scripts that are performed in your organization? What tools, techniques, approaches, and methods support this process?
7) How is the performance testing load generation process performed in your organization? What tools, techniques, and methods support this process?
8) How is the process of performance testing monitoring performed in your organization? What tools, techniques, and methods support this process?
9) What are the main monitored performance testing metrics in your organization's software projects?
10) How is the performance testing analyzing process performed from collected metrics in your organization?
11) What are the limitations found in the tools, methodologies, techniques, or methods that support performance testing activities in your organization?
12) Please provide in the field below any relevant considerations regarding this questionnaire or our research.
13) Which domains are subjected to performance testing in your organization?

We provide a help text for each question, to explain to the respondent which is more relevant in the respective question.

[8] In this question, we seek to understand each organization's maturity level for each performance testing activity and phase.

5.1.5 Pilot Questionnaire

Once the artifacts for conducting the survey are formulated, they must be validated to look out for bugs identification and emerging improvements in the instrument. Thereby, a pilot test should be conducted with a small sample similar to the target sample of the study so that these validations are performed before the survey execution. We intend to realize a pilot test with three (3) subjects, which were chosen because they have similar characteristics to our target sample.

5.1.6 Distribute the Questionnaire

The questionnaire should be distributed to selected members of the target audience as defined by the sampling plan. Social posts were refreshed or updated 3 times a week, always looking for the top of the feed to increase views and consequent survey adherence rate.

Regarding the WhatsApp group planned for submission, the request for the survey was sent, but it was considered inappropriate to the group's privacy standards, which according to their administrators is focused on informal information about performance testing.

5.1.7 Analyze the Results and Write a Report

The results should be collected and translated into appropriate graphical displays that facilitate understanding. The charts may be compiled into a report and interpretations, inferences, generalizations, and caveats can be made based on evidence provided by the results.

5.2 Threats to Validity

In this section, we introduce the main threats to the validity of our study, and strategies for mitigating these threats are presented according to Wohlin et al. (2012).

Construct Validity: Some threats may affect the validity of the constructor and are related to the possibility of its generalization. All artifacts produced we validated by two (2) performance testing experts to mitigate threats related to the construct validity.

Internal Validity: Threats to internal validity are forces that may alter the independent variable. For the elaboration of the questionnaire, the questions were grouped by similarity, and different strategies were used (open and closed questions). Also, we synthesize issues to avoid participants' fatigue as much as possible.

External Validity: External threats can limit our ability to generalize the results of the experiment externally. To mitigate threats related to the sample size, so that we could find a meaningful sample for our context, different means of participants prospecting were applied, such as the different social networks are chosen and the direct contact with IT companies.

Conclusion Validity: This type of threat is related to issues that may affect the correct conclusions based on the relationship between treatments and the results of the experiment. To mitigate the cultural bias inherent in surveys, we sought to reach the largest number of respondents per country, enabling us to increase confidence in the cultural data representativeness.

5.3 Results

The questionnaire was available from December 2019 to March 2020. Eighty nine (89) participants answered the questionnaire, twenty eight (28) of which had enough content, which gives us a utilization rate of approximately 31%. This relatively low percentage of adhesion, we attribute to the voluntary nature and questions being mostly optional. So it was a risk that we needed to take to try to extract as much information as possible.

5.3.1 Job Position

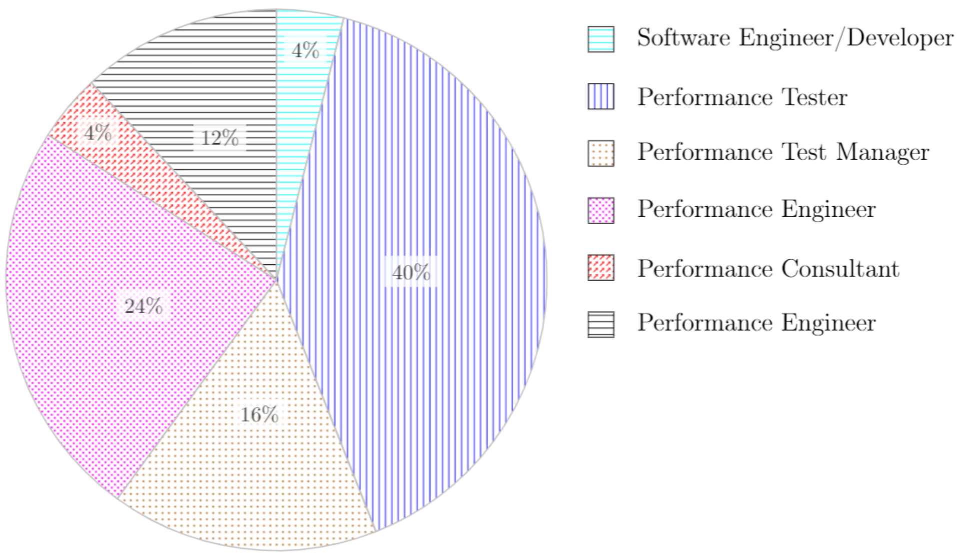
Figure 12 shows the percentages related to the subjects' positions in their respective organizations. According to these data, it is evident that a significant portion (40%) performs the function of Performance Tester, that is, it works directly with activities of test execution. Another relevant part of this graph is composed of Performance Engineers (16%), followed by 14% who declared to work as Performance Testing Manager, roles that have a broader participation in the entire test cycle. The job positions that had the least mention were Software Engineer and Performance Consultant, both with 4%, and Performance Analyst with 12%.

This information shows us first that the positions informed in the question correspond to the real scenario of roles played by the subjects when performing performance testing activities. Another piece of information that we can infer from the graph is that there is a significant portion of those directly involved in the activities of carrying out the test, which somehow qualifies our sample since we can obtain relevant information from an experienced target audience.

5.3.2 Subjects Experience

Figure 13 presents data related to the subjects' experiences. The graph shows that relating to the experience in the academic field, 48% have experience from 0 to 1 year, 20% from 2 to 4 years, 12% from 5 to 7 years, 8% from 8 to 10 years, 8% from 11 to 13 years, and 4% has more than 14 years experience in this area. On the other hand, the experience in the industry area presents other data. 4% have experience from 0 to 1 year,

Figure 12 – Job position.

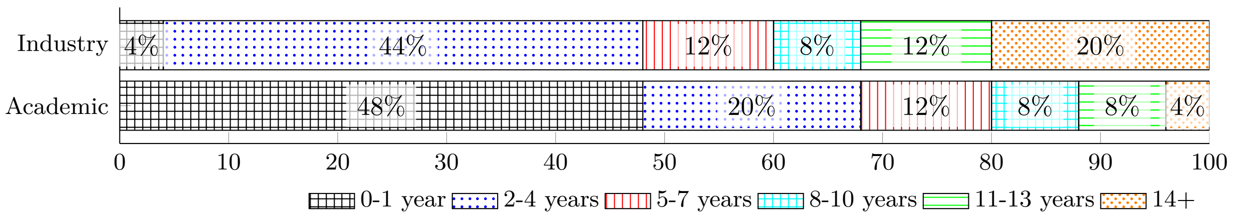


Source – Author.

44% have experience from 2 to 4 years, 12% from 5 to 7 years, 8% from 8 to 10 years, 12% from 11 to 13 years, and 20% reported more experience than 14 years in the field. Based on these data, we verified that approximately 40% of the participants have eight (8) years or more of professional experience.

These numbers show that the majority of the subjects have greater experience in the industry than in the academic area. This is relevant to our context since we seek practical data related to Performance Testing. In this case, it is more relevant than only academic experience.

Figure 13 – Subjects experience.



Source – Author.

5.3.3 Maturity Level

Meier et al. (2007) introduced Core Performance Testing Activities:

Identify Performance Testing Requirements (Requirements): Identify the physical test environment and the production environment as well as the tools and resources available to the test team. Identify the response time, throughput, and resource utilization goals and constraints. Additionally, identify project success criteria that may not be captured by those goals and constraints; for example, using performance tests to evaluate what combination of configuration settings will result in the most desirable performance characteristics.

Plan and Design Tests (Design): Identify key scenarios, determine variability among representative users, and how to simulate that variability, define test data, and establish performance metrics. Consolidate this information into one or more models of the system for implementation, execution, and analysis.

Configure the Test Environment (Configuration): Prepare the test environment, tools, and resources necessary to execute each strategy in a way that features and components become available for testing. Ensure that the test environment is instrumented for resource monitoring as necessary.

Implement Test Design (Implementation): Develop the performance tests following the test design.

Execution and Monitoring Tests (Execution): Run and monitor your tests. Validate the tests, test data, and results collection. Execute validated tests for analysis while monitoring the test and the test environment.

Analyze Results, Report, and Retest (Report): Consolidate and share results data. Analyze the data both individually and as a cross-functional team. Reprioritize the remaining tests and re-execute them as needed. When all of the metric values are within accepted limits, none of the set thresholds have been violated, and all of the desired information has been collected, you have finished testing that particular scenario on that particular configuration.

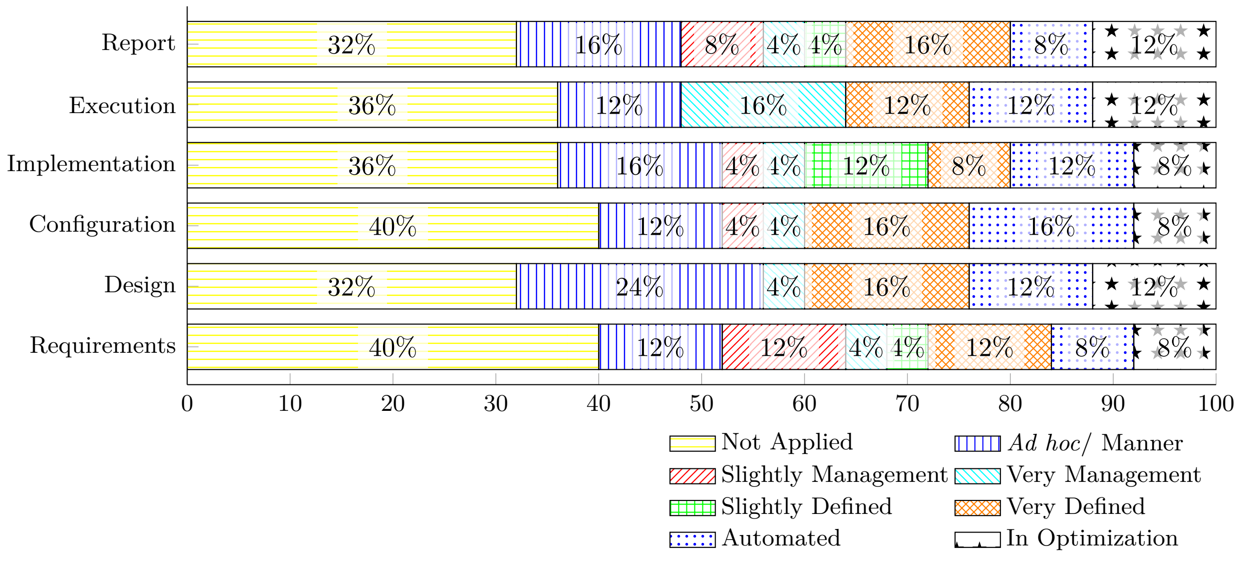
Based on these activities, we seek to obtain the maturity level according to the following scale. It is worth noting that the scale shown is increasing from 0-7 where each higher level contemplates the characteristics of the previous levels.

- **(7) In Optimization:** The activity is subject to continuous improvement;
- **(6) Automated:** The activity or some activity step/task is conducted in an automated manner;
- **(5) Very Defined:** The activity is conducted following documentation;
- **(4) Slightly Defined:** Some step/task of activity is defined through some documentation;
- **(3) Very Management:** The activity is conducted under management;
- **(2) Slightly Management:** There is some management in activity execution;
- **(1) *Ad hoc*/ Manner:** The activity is applied in an *ad hoc* manner;

- **(0) Not Applied:** The activity is not applied.

Figure 14 details the information we obtained about the maturity level of the performance testing process in the subjects organizations. An interesting piece of data that we can highlight in this graph is that approximately 51.33% of the participants informed that they do not have a defined process, namely, they do not apply or apply in an *ad hoc* manner. This behavior is common to all phases of performance testing. So, we ratify our motivation to propose a process that assists those involved in performance testing in the planning, design, and execution of the test. We believe that a generic framework may contribute to organizations applying a systematic and well-defined process for the test performance, as well as to provide a repository of relevant information about this type of test.

Figure 14 – Maturity level.



Source – Author.

5.4 Chapter Summary

In this chapter, we presented the details of the protocol, execution, and results of the survey that we applied. This survey aimed to obtain subsidies for the preparation of PTBOK from the industry perspective. The main difficulty that we faced in conducting this study was the participant's adhesion. Furthermore, obtaining answers provided some insights for the PTBOK. However, the results that we found in the survey converge to the SLR data.

6 PTBOK

This chapter is organized as follows. In Section 6.1 we present an overview of the PTBOK framework. Sections 6.2, 6.3, 6.4, 6.5, 6.6 and 6.7 detail the main PTBOK components. In Section 6.9 we show some details of the PTBOK modeling.

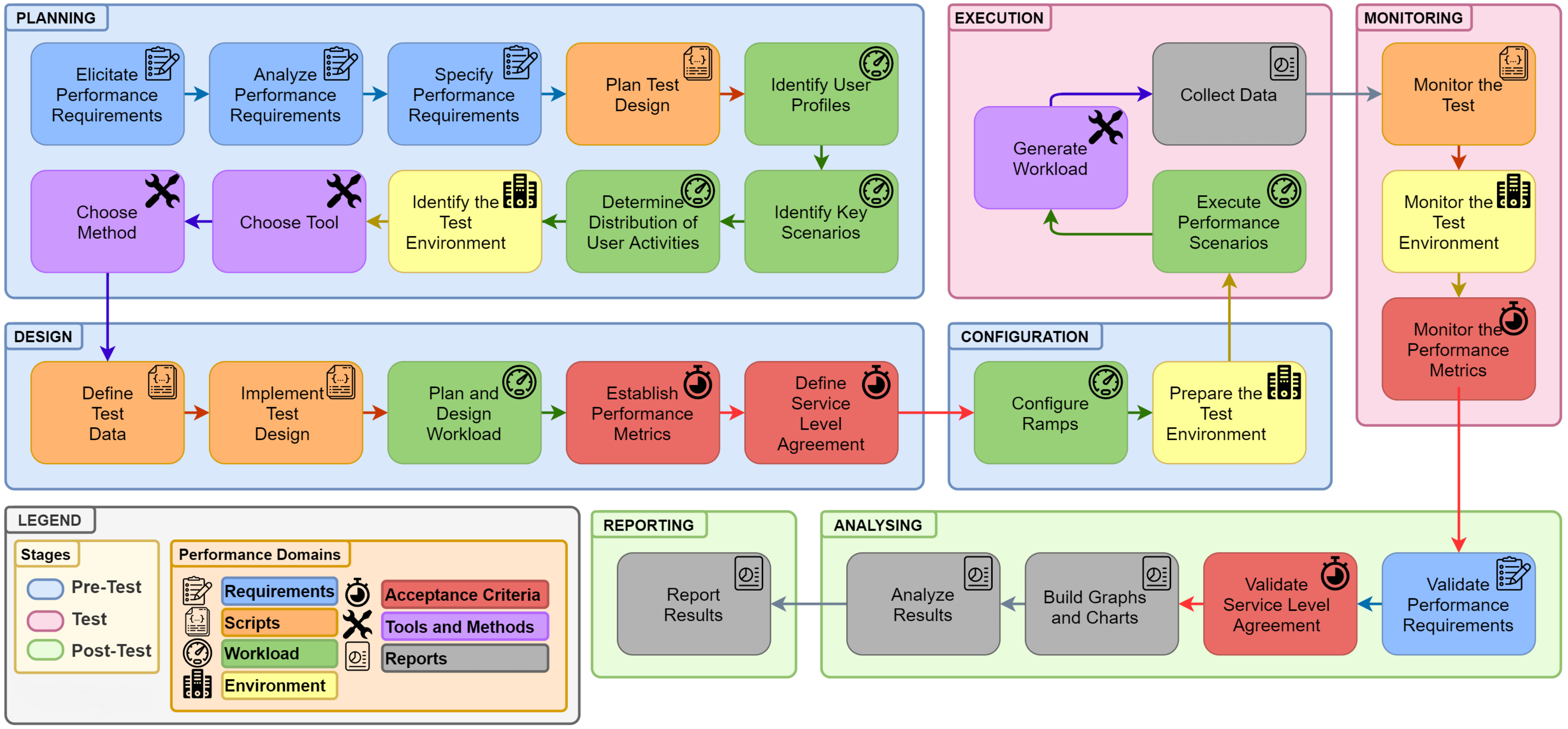
Performance Testing Body of Knowledge (PTBOK) gathers the main inputs and outputs, techniques, methods, approaches, *etc.* related to Performance Testing. Also, PTBOK seeks to guide, from a practical as well theoretical view, the entire performance testing process, involving a set of domains, stages, phases, activities, and steps for conducting the Performance Testing.

6.1 Framework

In the following, we describe PTBOK components. A Stage is at the highest abstraction level of the process and is composed of Phases (Planning, Design, Configuration, Execution, Monitoring, Analyzing, and Reporting), which in turn are composed of activities. Each Activity may have one or more tasks. A Task is performed by a Role, that is the Actor responsible for performing the referred task, and these Roles may be defined in a Role Set, which aggregates a set of roles with similar functions. In another point of view, Performance Domains are areas of knowledge related to a given domain. In the framework, it is also possible to visualize this relationship between Performance Domains and the activities related to them through a color pattern.

Figure 15 represents the PTBOK framework, which shows an overview of the process. Each phase is grouped with its respective activities. Furthermore, through a color pattern, one can identify the relationship amongst activities vs stages and activities vs performance domains.

Figure 15 – PTBOK framework overview



Source – Author.

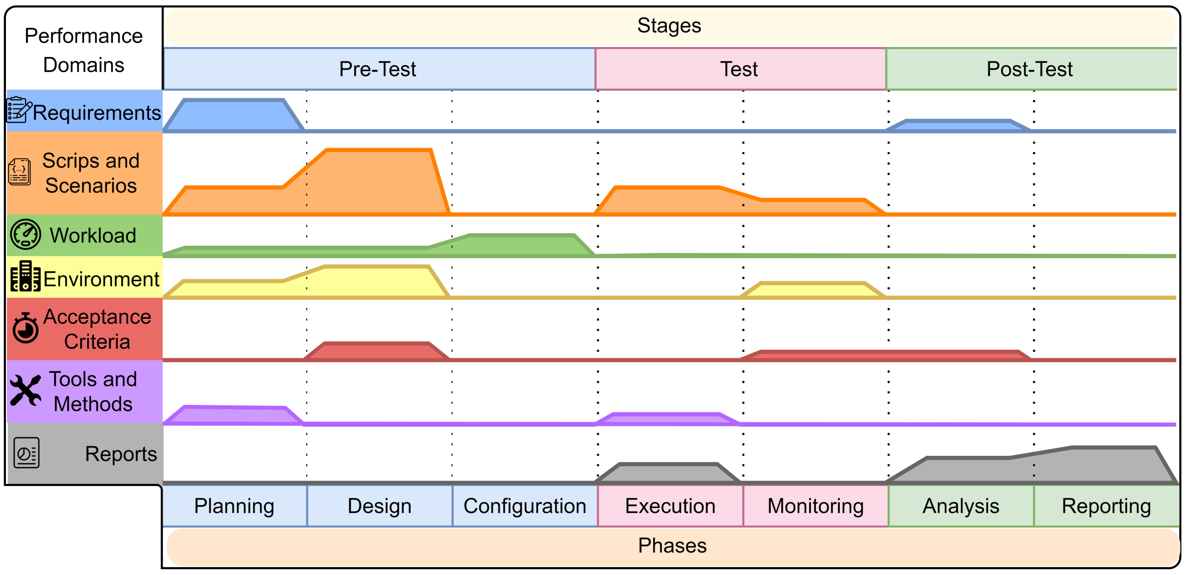
6.1.1 Effort Analysis

In Figure 16, we intend to represent an analysis of the effort employed in each of the phases and stages of PTBOK according to the related Performance Domains.

To obtain the values shown in Figure 16, we carried out an *ad hoc* study in our research group. This analysis took into account the number of suggested PTBOK activities in each phase and their complexity. We emphasize that these values are not absolute and may change according to how each organization conducts performance testing. In addition, we analyzed this figure in the survey for PTBOK evaluation, aiming to ratify the values shown in the graph with expert opinions.

Regarding Requirements, most of the work is done in the Planning phase, and there is also a relationship with the Analysis phase. Scripts and Scenarios have greater use in the Project phase, but they also appear in the Planning and also in the test stage, through the Execution and Monitoring phases. Workload appears only in the Pre-Test stage, most frequently in the Configuration phase. The Performance Domain Environment is used in the Planning and Design phases but is also related to Monitoring. Acceptance Criteria is used in the three stages, through the Design, Monitoring, and Analysis phases. Tools and Methods has slightly greater use in Planning and it is also listed in the Execution phase Finally, Performance Domain Reports has a little effort in the Execution phase, but the vast majority of work in this domain is in the Analysis and mainly Reporting phases

Figure 16 – PTBOK estimated effort: Performance Domains vs Phases vs Stages



Source – Author.

6.2 Role Set

In some cases, specific roles are not necessary for performance testing, it depends on how the roles are structured in the organization. Thus, it is possible to adhere to a more generic representation through a Role Set. A Role Set can be defined as a grouping of responsibilities in a performance testing context.

- **Performance Engineer**

This Role Set gathers skills and competencies to perform throughout the performance testing life cycle. Given the diversity of tasks and contexts in which the Performance Engineer may act, it was specialized in 3 (three) different roles: Performance Analyst, Performance Architect, and Performance Tester. However, these specializations are not mandatory, and it is the responsibility of the organization to map needs for an appropriate Performance Engineer decomposition.

6.3 Roles

The following are the main roles involved in the performance testing phases. We define a schema for the prefix of each activity. The letter represents the phases **P**lanning, **D**esign, **C**onfiguration, **E**xecution, **M**onitoring, **A**nalysing, and **R**eporting. The first number represents Performance Domain:

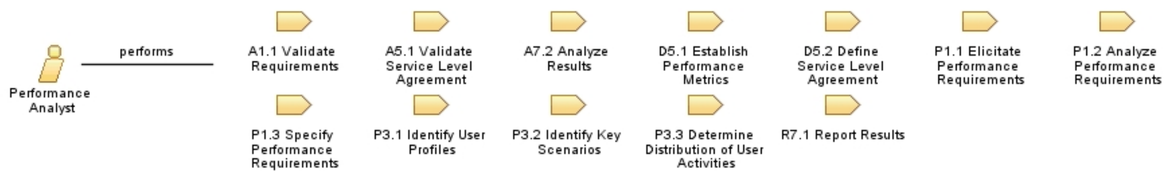
1. Requirements
2. Scripts
3. Workload
4. Environment
5. Acceptance Criteria
6. Tools and Methods
7. Reports

The second number represents the order in which the activity takes place according to the context of the Performance Domain and Phase. For example, activity P1.1 belongs to the Planning phase, and it is the first activity of Performance Domain Requirements.

6.3.1 Performance Analyst

Performance Analyst participates in early and late performance testing phases. It is responsible for initial testing planning and documentation, providing input to subsequent phases, design, and configuration. This role is also observed after testing execution, on this account, in the analysis and reporting phases. Figure 17 shows the activities performed by this role.

Figure 17 – Performance Analyst Activities

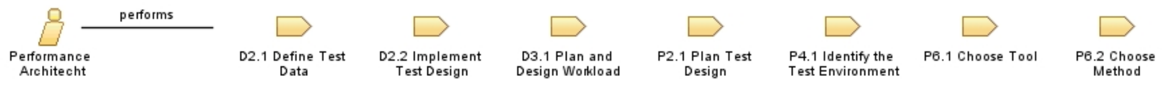

performs
Performance Analyst
A1.1 Validate Requirements
A5.1 Validate Service Level Agreement
A7.2 Analyze Results
D5.1 Establish Performance Metrics
D5.2 Define Service Level Agreement
P1.1 Elicitate Performance Requirements
P1.2 Analyze Performance Requirements
P1.3 Specify Performance Requirements
P3.1 Identify User Profiles
P3.2 Identify Key Scenarios
P3.3 Determine Distribution of User Activities
R7.1 Report Results

Source – Author.

6.3.2 Performance Architect

This role is involved within Design and Configuration Phases and it is responsible for bridging early phases and testing execution. A Performance Architect must have skills in activity designing and configuration. Figure 18 shows the activities performed by this role.

Figure 18 – Performance Architect Activities



Source – Author.

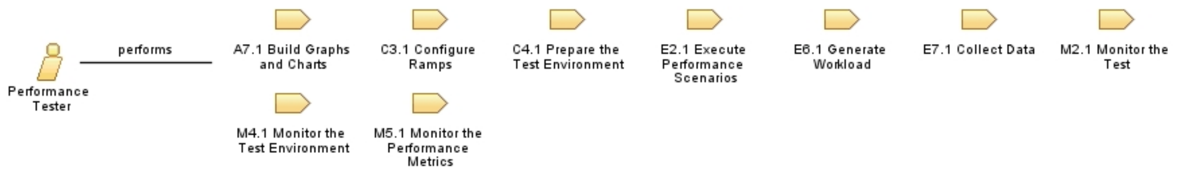
6.3.3 Performance Tester

This role is directly related to the testing execution phase. A Performance Tester is the one who should run performance testing, making use of available tools for performance testing. Figure 19 shows the activities performed by this role.

6.4 Stages

A performance test has three (3) main stages, the first precedes the performance test execution activities (Pre-Test stage). The second gather the activities carried out

Figure 19 – Performance Tester Activities


performs
Performance Tester
A7.1 Build Graphs and Charts
C3.1 Configure Ramps
C4.1 Prepare the Test Environment
E2.1 Execute Performance Scenarios
E6.1 Generate Workload
E7.1 Collect Data
M2.1 Monitor the Test
M4.1 Monitor the Test Environment
M5.1 Monitor the Performance Metrics

Source – Author.

in the performance testing execution (Test stage). Finally, activities performed after test execution are carried out in the Post-Test stage. We emphasize that all stages are complementary to each other and necessary since each of these stages brings together essential characteristics for performance testing. A relevant issue concerning the stages is the execution flow, that is, in the way we present, the stages must occur sequentially since the output from one stage is an input to the subsequent stage.

6.4.1 Pre-Test

Pre-Test comprises the initial activities, those related to the performance testing planning, definition, and preparation.

Pre-Test stage has three (3) phases: Planning, Design, and Configuration. Pre-Test stage is characterized by the performance of activities and tasks that precede the test execution. This stage has a main appeal to questions related to the test preparation, providing from its outputs, inputs for the Test stage.

6.4.2 Test

The Test stage occurs from the outputs of the previous stage (pre-test) and consists of two (2) phases, Execution, and Monitoring. The execution phase brings together the activities related to the workload application according to the planned scenarios. The Monitoring phase, in turn, is responsible for verifying the application's behavior as a result of the applied workload.

6.4.3 Post-Test

Post-Test stage is the last performance testing stage. Analyzing and Reporting phases to compose this stage, which is executed from the Test stage outputs. The first phase of this stage (Analyzing) has as main objective to perform an analysis on the data generated from the defined metrics such as throughput, latency, memory consumption, processing, etc. From this analysis, it is possible to present the final data related to the test, through reports in the Reporting phase.

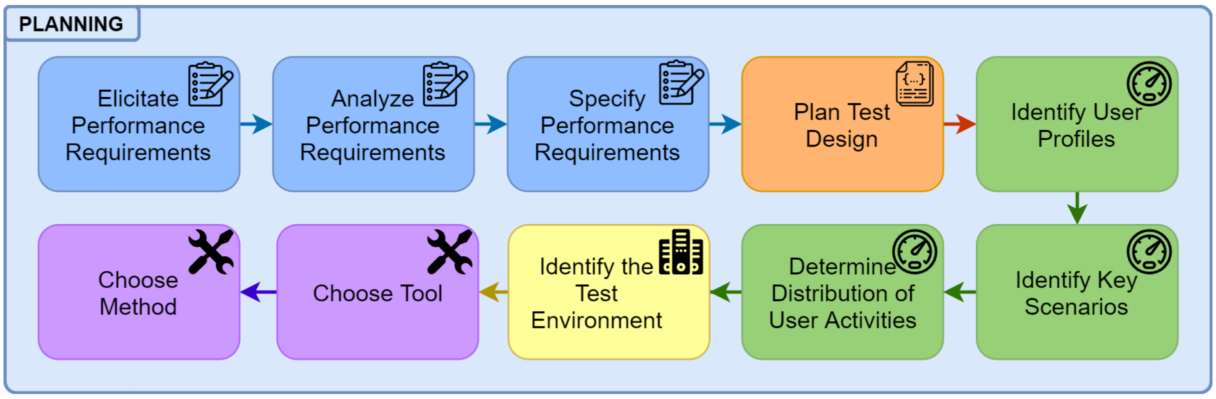
6.5 Phases

We mapped seven (7) main phases for carrying out performance testing: Planning, Design, Configuration, Execution, Monitoring, Analysis, and Reporting. These phases are described as follows.

6.5.1 Planning

The planning phase is the first performance testing phase. In this phase, some factors are mapped, such as the allocation of resources, infrastructure, data, and also human resources involved in the test. The general scope of the performance test should be defined during the planning phase. In addition, the identification and analysis of risks, as well as information relevant to the test, should also be mapped. In summary, the definition of the test occurs in the planning phase. Figure 20 shows the activities performed in this phase.

Figure 20 – Planning Activities

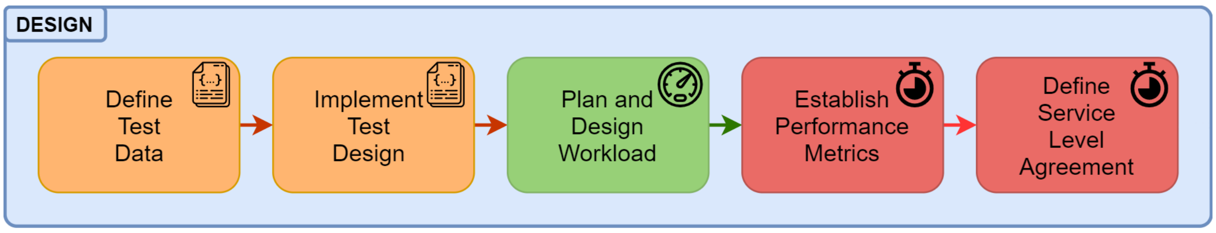

PLANNING
Elicitate Performance Requirements
Analyze Performance Requirements
Specify Performance Requirements
Plan Test Design
Identify User Profiles
Identify Key Scenarios
Determine Distribution of User Activities
Identify the Test Environment
Choose Tool
Choose Method

Source – Author.

6.5.2 Design

Using the test specifications defined in the planning phase, performance testing is designed taking into account environments' particularities and performance testing goals. This phase includes activities related to the test project, receiving inputs from the planning phase, and providing outputs to the execution phase. The Design phase also involves activities focusing on script elaboration, which can be gathered by different means. For instance, supported by models and using the Model-Based Testing or Capture and Replay for an automatic generation or by coding, where scripts are created from a specific programming language. Figure 21 shows the activities performed in this phase.

Figure 21 – Design Activities

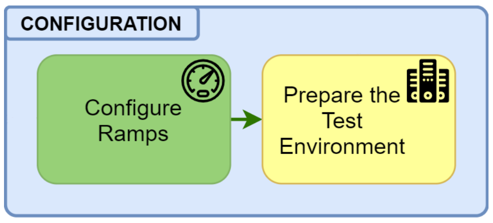


Source – Author.

6.5.3 Configuration

The main activities of performance testing depend fundamentally on automation tools for the realization of their tasks. Therefore, in the configuration phase, adjustments to the necessary tools occur. These adjustments include both the tools related to the test execution and the tools related to the environment where the test will be performed. It is the last phase before test execution. In this state, adjustments and setting performance testing are made. Issues like workload type, performance testing type and tool functionalities should be considered as well as infrastructure issues. Figure 22 shows the activities performed in this phase.

Figure 22 – Configuration Activities

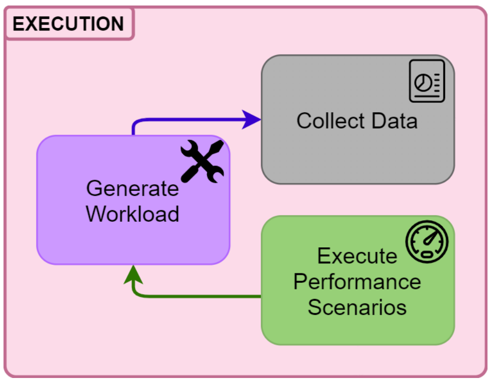


Source – Author.

6.5.4 Execution

In the execution phase, the workload is applied to SUT. This phase characterizes performance testing since it is essential for any performance testing to generate and apply a certain workload in an application. The activities carried out in this phase depend on tools that automate their tasks, because it is impracticable to perform tests manually, that is, without the support of tools that automate and facilitate this execution. It is worth emphasizing that this phase must provide mechanisms that allow monitoring of the metrics defined in the early phases. Figure 23 shows the activities performed in this phase.

Figure 23 – Execution Activities



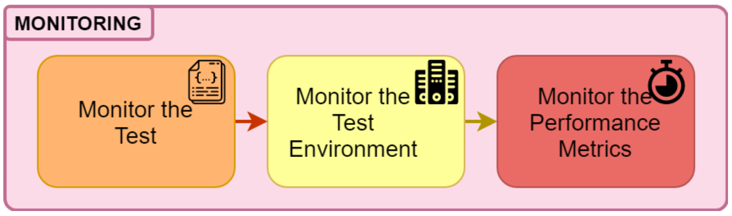
Source – Author.

6.5.5 Monitoring

Defined metrics as throughput, response time, hits per second must be monitored during test execution as well as bottlenecks identification. This monitoring allows performance testing roles to obtain outputs for subsequent phases, analysis, and reporting in the post-test stage.

The monitoring phase has a strong connection with the execution phase, in some cases, they can be confused, because some tools execute and monitor simultaneously. However, we find it interesting to have this separation due to the degree of importance that both phases have in the performance test. Figure 24 shows the activities performed in this phase.

Figure 24 – Monitoring Activities

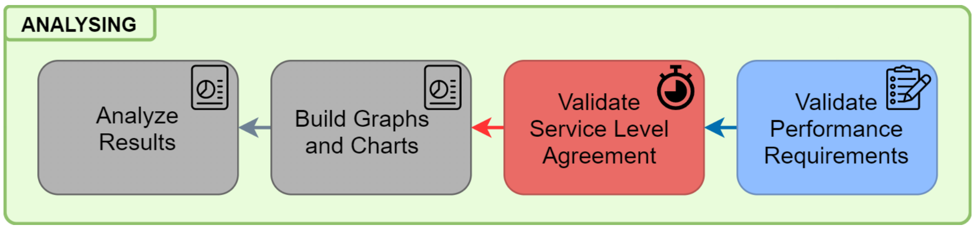


Source – Author.

6.5.6 Analysis

This phase refers to how data results from the monitoring are processed and represented in performance testing reports. Hence, data must first be compared with the test objective, which is a success criterion in test analysis. By understanding the SUT behavior, it is possible to draw conclusions based on the data so that reports are generated in the next phase. In addition, in the analysis phase, possible actions can be mapped to correct bottlenecks and unexpected application behaviors. Data should be presented as representative as possible, either through spreadsheets, tables, or charts, the latter facilitating the identification of trends. Figure 25 shows the activities performed in this phase.

Figure 25 – Analysis Activities

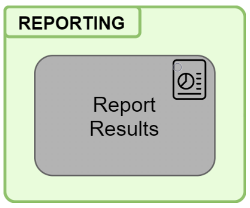

ANALYSING
Analyze Results
Build Graphs and Charts
Validate Service Level Agreement
Validate Performance Requirements

Source – Author.

6.5.7 Reporting

The test reports are generated in the Reporting phase, with the objective of providing an understanding of the possible bottlenecks detected in the test and also offering the test results in a legible and organized manner. The reporting phase is decisive for an understanding of the application's behavior after the test execution. The monitored metrics must be detailed through robust documentation that allows the Performance Engineer to explain to stakeholders the test results as well as the possible insights generated from performance testing. Figure 26 shows the activity performed in this phase.

Figure 26 – Reporting Activities



Source – Author.

6.6 Performance Domains

The following are the 7 (seven) performance domains related to performance testing.

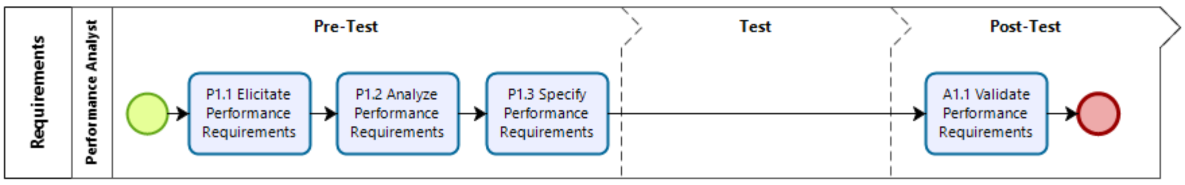
6.6.1 Requirements

To obtain a clear definition of the performance testing requirements, an understanding of the functionalities is necessary. Thus, the Performance Engineer can make use of known requirements elicitation techniques, since there is no clear understanding of what features should be tested. However, in a scenario where the main functionalities for testing are known, it is up to the Performance Analyst, during the Project phase, to define the main inputs and expected results, as well as issues related to Workload, Ramp Up and Ramp Down. Documenting requirements through models is important to ensure that the application's test history is available, thus seeking greater ease in the evolution and replication of application performance tests.

This performance domain comprises information that is directly related to the test requirements. The activities included in this performance domain involve requirements elicitation, analysis, and validation.

Figure 27, shows the related activities to this performance domain, as well as the stage in which they occur and which role is responsible for which activity.

Figure 27 – Requirements Activities


Requirements
Performance Analyst
Pre-Test
Test
Post-Test
P1.1 Elicitate Performance Requirements
P1.2 Analyze Performance Requirements
P1.3 Specify Performance Requirements
A1.1 Validate Performance Requirements

Source – Author.

- **P1.1 Elicitate Performance Requirements**

Different sources may provide information related to the test, for example, the application development and maintenance team and stakeholders. Therefore, it is necessary to collect, compile, and save this information in the requirements specification document. Requirements elicitation techniques may support this activity.

Steps:

1. Gathering performance testing info;
2. Compiling performance testing info;
3. Updating Performance Testing Requirements Specification

   - **Added Elements**: SUT description; environment description; contractual obligations; client expectations; mitigation of risks; business requirements; available application features and components; application usage scenarios

- **P1.2 Analyze Performance Requirements**

This activity aims to perform the analysis of the test requirements and it has an approach similar to that used in traditional Requirements Engineering. Therefore, requirements analysis techniques may support this activity. During the analysis of the test requirements, the following information should be added to the Performance Testing Requirements Specification: Objectives (detailing the general objectives of the test); Targets (desired values to metrics); thresholds (max limit acceptable).

- **P1.3 Specify Performance Requirements**

This activity consolidates the elicitation and analysis of the test requirements. It is important to note that these three activities (P1.1, P1.2, and P1.3) are closely linked and, depending on the context of the organization, may be carried out simultaneously. Therefore, the requirements document must be revised and corrected, whether necessary. In this activity, it is added the feasibility analysis to the performance testing requirements specification: are the goals plausible and achievable for the organization's scenario?

- **A1.1 Validate Performance Requirements**

This is the first analysis activity in the process and aims to validate the requirements. So, one can use V&V techniques from Software Engineering, for example, reviews, walkthroughs, and inspections, to assist in the validation process. One must update the Performance Testing Requirements Specification with the validation data after applying the V&V technique.

6.6.2 Scripts and Scenarios

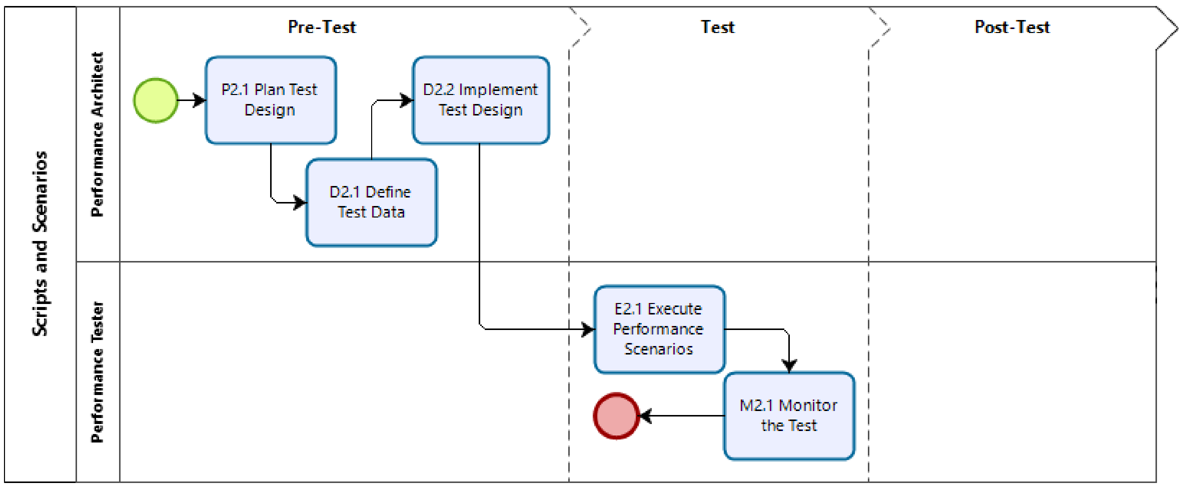
In general, a script can be defined as a sequence of steps necessary to carry out a performance testing execution instance. Scripts can be developed through programming languages, such as C, Java, Python, etc. Another manner of obtaining scripts is through Capture & Replay tools, which execute a certain functionality of the application to generate the scripts from that execution. Hence, it can imply a decrease in the level of complexity for the elaboration of the scripts, the code is abstracted from the user. However, this approach can generate excess code or even some type of undesired code. Therefore, despite a higher level of complexity, the manual script preparation can be very useful in a context where the team is proficient in the scripts language or even in cases where script changes are constantly needed. An important aspect is in relation to the creation of the scripts: is the possibility of scripts parametrization, that is, where through external files, the script execution parameters are dynamically changed. Without this kind of functionality, performance testing becomes somewhat impractical. The most available performance testing tools already allow the use of both, manual scripts and Capture & Replay approaches, as well as the scripts parametrization.

A scenario can be defined as a specific context, where application functionality will be tested. This context is composed of the workload and the way that this load behaves, and also by the set of all characteristics inherent to that context. For example, a load of 1000 initial users being incremented every 2 minutes in a product's purchase functionality through an application's shopping cart. This scenario is formed by a set of one or more scripts that will be executed so that the projected objective is reached. The scenarios must be thought of in the initial phases of the test, more specifically in the planning and in the requirements domain, so that later in the phases of design and configuration this scenario is instantiated for its execution by scripts.

Below we detail the main activities of this performance domain. The execution of the test is the main focus of the activities of this domain.

Figure 28 shows the related activities to this performance domain, as well as the stage in which they occur and which role is responsible for which activity.

Figure 28 – Scripts and Scenarios Activities



Source – Author.

- **P2.1 Plan Test Design**

This activity is the first project activity in the process. The main project inputs are the test objectives and the SUT information, which may be found in the Performance Testing Requirements Specification. After extracting this information, there are already subsidies for the creation of a Performance Testing Plan. This initial document may contain some preliminary design information, evolved from the test specifications, such as the intended approach, resource allocation, strategies, etc. However, the test project is still immature at this stage, so there is no need for detailed project detail at this moment.

- **D2.1 Define Test Data**

As the first design activity, it occurs the test data definition. Then, the type of approach is selected, for example, random or database-based, for data acquisition. After that, one performs the test data definition and the execution plan included in the Performance Testing Plan. Therefore, they should reflect inputs as close as possible to the application production environment.

- **D2.2 Implement Test Design**

This activity has a relatively simple definition, summarizing the creation of the scripts and models used in the test. Then, one must follow the planning. The tool that was chosen to create the models and scripts and perform the test directly influences this activity.

- **E2.1 Execute Performance Scenarios**

The first activity in the test execution is the execution of the scenarios, which depends fundamentally on the tool and approaches chosen for the test. After running the scenarios, one record scenarios information in the Performance Testing Plan.

- **M2.1 Monitor the Test**

Test monitoring is divided into three main activities. The first is the monitoring of the test. This activity has a higher-level view of the test, analyzing the test execution in general, and recording the monitoring data in the Performance Testing Report and Performance Testing Plan.

6.6.3 Workload

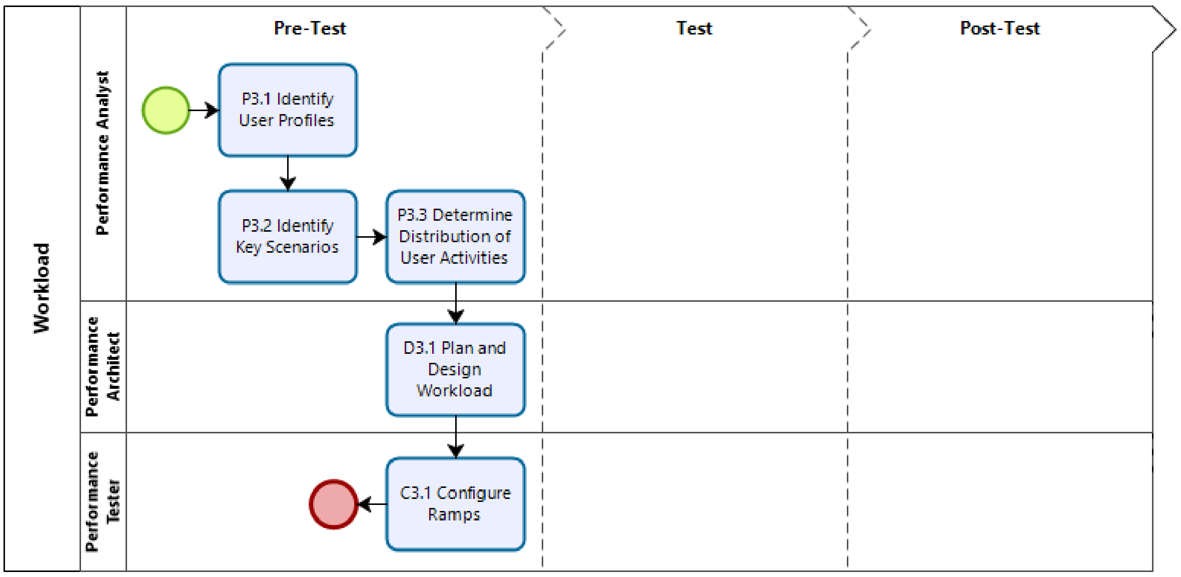
The Workload domain covers all aspects related to the generation, execution, and monitoring of the workload, with some important tasks like identifying key scenarios, determining variability among representative users and how to simulate that variability, defining test data, and establishing metrics. Consolidating this information into one or more system usage models.

Some relevant information, such as the number of virtual users and the delay in user actions, should be considered in the workload modeling. However, the system flow and throughput are directly affected by the processing time, which can be variable as the workload increases.

The planning and correct execution of the Workload is fundamental to the success of the test. Therefore, below we detail the activities present in this performance domain.

Figure 29, shows the related activities to this performance domain, as well as the stage in which they occur and which role is responsible for which activity.

Figure 29 – Workload Activities


Workload
Performance Analyst
Performance Architect
Performance Tester
Pre-Test
Test
Post-Test
P3.1 Identify User Profiles
P3.2 Identify Key Scenarios
P3.3 Determine Distribution of User Activities
D3.1 Plan and Design Workload
C3.1 Configure Ramps

Source – Author.

- **P3.1 Identify User Profiles**

The test should be performed in an environment as close as possible to the production environment. Thus, the different user profiles that interact with the application must be identified. Therefore, this activity aims to map these profiles.

- **P3.2 Identify Key Scenarios**

Mapping the main usage scenarios of the application is essential for proper test execution. A key scenario has some peculiarities that characterize it, gathering access characteristics of a user profile with the related functionalities, as well as the flow in which they occur. In this activity, the main scenarios are extracted and the Performance Testing Plan is updated with this information.

- **P3.3 Determine Distribution of User Activities**

After obtaining the main user profiles and key scenarios, it is necessary to make a relationship between them. So, profiles are allocated to the scenarios, and the Performance Testing Plan is implemented with this information.

- **D3.1 Plan and Design Workload**

It is important to map the way that the workload behaves according to the used approach (load, stress, volume, etc.). To define the workload project since each of these approaches has particularities that influence this behavior. Thus, in this activity, the Design Workload is defined and elaborated and must be attached to the Performance Testing Plan.

- **C3.1 Configure Ramps**

Before the test run, we need to perform some configuration activities before running the test. The first is to configure the workload ramp behavior. This ramp has three main phases, ramp up, its steady state, and its ramp down. Each of these phases may happen in different ways according to the test approach. Typically, the tools used for the test run offer customized settings for the ramp configuration.

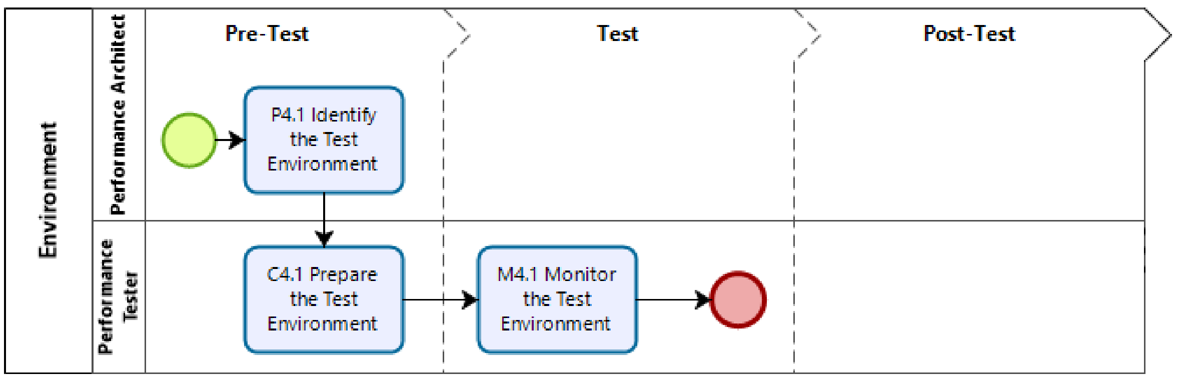
6.6.4 Environment

In the performance testing context, the environment domain covers everything related to the infrastructure used in the test, both hardware or software. Thus, some key activities related to this domain must be carried out while preparing the test environment, tools, and resources needed to execute each strategy as resources and components become available for testing, ensuring that the test environment is instrumented for monitoring resources as needed.

Below, we detail the activities of this performance domain

Figure 30, shows the related activities to this performance domain, as well as the stage in which they occur and which role is responsible for which activity.

Figure 30 – Environment Activities


Environment
Performance Architect
Performance Tester
Pre-Test
Test
Post-Test
P4.1 Identify the Test Environment
C4.1 Prepare the Test Environment
M4.1 Monitor the Test Environment

Source – Author.

- **P4.1 Identify the Test Environment**

Environment information is analyzed in this activity. This includes both information about the application's production environment and information about the test environment that will be used in the test run. Environment planning includes possible requests for resources that can be made. Thus, in addition to environmental planning, the Testing Environment Request should be added to the Performance Testing Plan.

- **C4.1 Prepare the Test Environment**

The second and last configuration activity is preparing the test environment. Then, we must set up the tools and artifacts for the test execution. Configurations of the SUT and also of the server, network, etc., are performed in this activity.

- **M4.1 Monitor the Test Environment**

This is the second monitoring activity and it aims to monitor the testing environment. As in the test monitoring, we must record the information in two artifacts, in the Performance Testing Plan and Performance Testing Report the environment status info, and the environment monitoring info, respectively.

6.6.5 Acceptance Criteria

This domain has some characteristics that are similar to those of the requirements domain. However, the main difference is that the acceptance criteria have a greater focus on performance testing business rules, with an approach more focused on the results and metrics, while the requirements domain looks at the application and lists the features to be tested.

The main activities in this domain involve identifying the response time, throughput, and resource utilization goals and constraints. In general, response time is a user concern, throughput is a business concern, and resource utilization is a system concern. Additionally, identify project success criteria that may not be captured by those goals and constraints; for example, using performance tests to evaluate what combination of settings will result in the most desirable performance characteristics.

Figure 31, shows the related activities to this performance domain, as well as the stage in which they occur and which role is responsible for which activity.

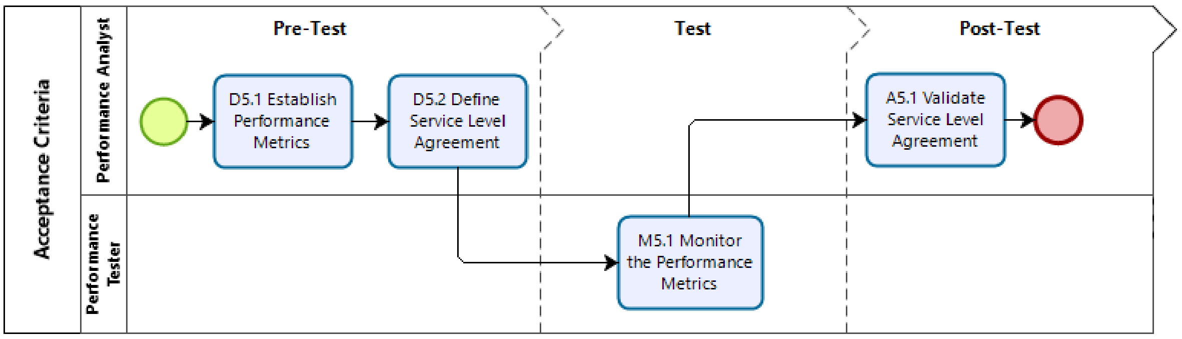
The activities of this performance domain are as follow.

- **D5.1 Establish Performance Metrics**

To carry out the activity that establishes the performance metrics, it is important to gather information about the metrics that will be collected during the test execution. These metrics fall into two categories, server and application metrics. After the definition, we need to create a baseline, which will establish the default values for the metrics according to the test requirements.

- **D5.2 Define Service Level Agreement**

Figure 31 – Acceptance Criteria Activities


Acceptance Criteria
Performance Analyst
Performance Tester
Pre-Test
Test
Post-Test
D5.1 Establish Performance Metrics
D5.2 Define Service Level Agreement
M5.1 Monitor the Performance Metrics
A5.1 Validate Service Level Agreement

Source – Author.

In summary, SLAs are specific goals that we set for the application. These goals are based on the best possible interaction with users regarding performance. To carry out this definition of SLAs, it is necessary to analyze the behavior of certain types of application usage and map out arisen bottlenecks that can impair the application's functioning. From these values, it is possible to define SLAs. After performing the test, we must compare these values with the values obtained during the test execution, indicating whether the application meets the planned agreements.

- **M5.1 Monitor the Performance Metrics**

This activity aims to monitor performance metrics. As with other monitoring activities, in addition to showing the information and alerts, it is necessary to record the monitored data in the Performance Testing Report.

- **A5.1 Validate Service Level Agreement**

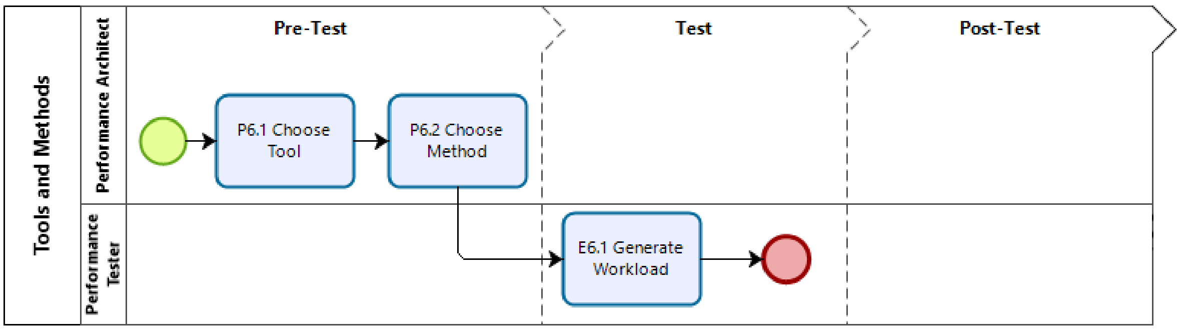
In this activity, we make a comparative analysis of the SLA with the metrics, intending to validate the SLAs. It is important to detail whether the SLAs are consistent with the test or the test is suitable for the SLAs. After performing this analysis, it is needed to record the data of this validation in the Performance Testing Report.

6.6.6 Tools and Methods

Tools are extremely important for performance testing since testing without tools to automate the process is impractical. Therefore, we have a specific domain that deals with the tools and methods used. Tools are available on the market for virtually the entire performance testing life cycle. As for the methods, some approaches to performance testing (load, stress, spike, etc.), as well as techniques (scripting, capture & replay), should be analyzed and chosen according to the test purpose and context.

Figure 32, shows the related activities to this performance domain, as well as the stage in which they occur and which role is responsible for which activity.

Figure 32 – Tools and Methods Activities



Source – Author.

- **P6.1 Choose Tool**

During planning, it is possible to choose tools to support the test. So, the Performance Architect must review the list of tools and choose the most appropriate for the context. After that, it is important to register in the Performance Testing Plan the justification and what requirements will be covered with the use of the tool.

- **P6.2 Choose Method**

Choosing the appropriate method for the test is an important task, as the method must meet what is expected for the test. Hence, we extract the necessary information from the specification document and then update the Performance Testing Plan. The list of approaches assists in the definition and choice of the method, as it brings the characteristics of each of the methods and approaches used in the performance test. This choice involves both issues of approaches (load, stress, spike, volume) as well as questions specific to the method such as MBT, CR, among others. This activity ends the test planning activities.

- **E6.1 Generate Workload**

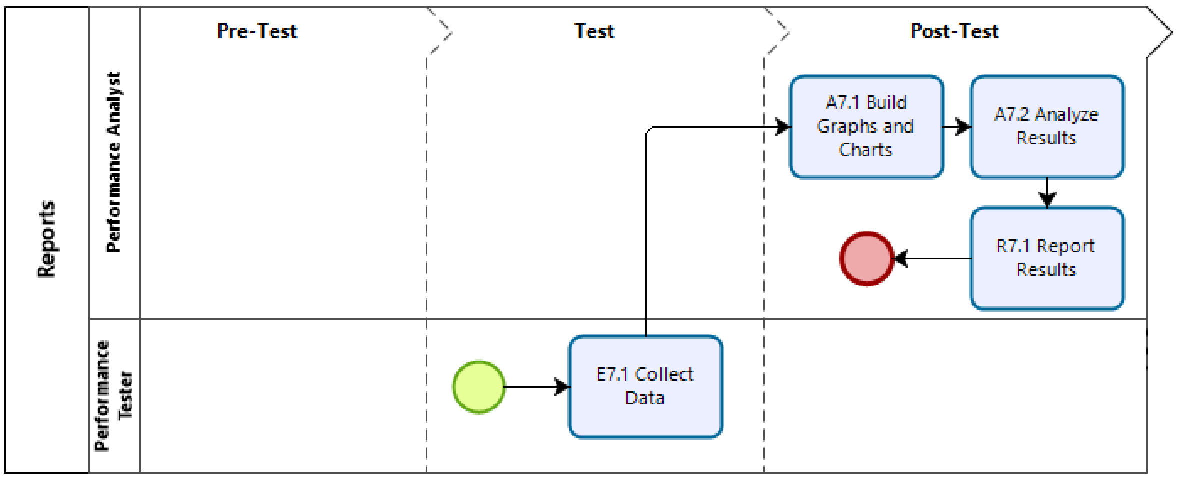
The basis of performance testing is workload generation. Thus, creating virtual users that interact with the application by simulating real users may contribute. This load generation is offered in an automated way by most tools, with the tester only being responsible for its configuration and management. It is important to record the workload status in the Performance Testing Plan.

6.6.7 Reports

The main objective of the reports is to consolidate and share the results obtained in the test, so they are considered important in that specific domain. Reports must be detailed enough to support the performance engineer in making the decision, which may include adaptations and repetitions in the test. A test that does not have reports with relevant information, which indicates the application state according to the workload performed, tends to have its usefulness compromised.

Figure 33, shows the related activities to this performance domain, as well as the stage in which they occur and which role is responsible for which activity.

Figure 33 – Reports Activities



Source – Author.

Below, we detail the activities of this performance domain.

- **E7.1 Collect Data**

After the beginning of the test run, we must collect data related to the metrics that were defined in the planning and in the Project. The use of specific tools is essential to carry out this activity. We record the metrics collected in the Performance Testing Report.

- **A7.1 Build Graphs and Charts**

The graphs and tables build are very important to show intuitively the data obtained in the test execution and consequently in the test reports. The vast majority of tools used to run the test support a considerable variety of graphs. In general, for each metric, it is possible to generate a graph or table or merge metrics in combined graphs. If the test execution tool does not support this type of representation, one can use external tools that enable this creation since the information present in the Performance Testing Report must be intuitive for the team.

- **A7.2 Analyze Results**

In this activity, one needs to analyze the results. Hence, the person responsible for the analysis should have some mathematical knowledge and statistical principles to interpret the test results. Some tools perform this analysis in an automated way or even assist the user in such analysis. However, despite the tool that helps a lot in this activity, the analysis by the professional of the area cannot be disregarded because it is through it that we may have a real understanding of the results obtained with the test.

- **R7.1 Report Results**

This activity ends the testing process and aims to report the results, which may be from two perspectives. The first is a technical vision that should provide useful evidence and information to the technical team involved in the test, developers, architects, software engineers in general. The second perspective focuses on stakeholders, customers in general, who are interested in a report focused on business rules information affected by the test. Three principles assist in the formulation of these reports: Report Early, that is, to make the reports as early as possible, Report Visually, offering information with visual appeal, and Report Intuitively, aiming to generate reports as intuitive as possible.

6.7 Artifacts

The most relevant artifacts in this context are as follows.

6.7.1 Performance Testing Requirements Specification

This artifact gathers the main information related to the test requirements. It is an incremental document, just like the Performance Test Plan, that is, it is incremented with information according to the target activities. The Performance Analyst is responsible for preparing this document. However, due to a large number of activities making use of it, some other roles such as Performance Architect and Performance Tester may also make use of the document's information, and also update it with information.

6.7.2 Performance Testing Plan

A Performance Testing Plan is a document elaborated by Performance Analyst as 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.

6.7.3 Model

This artifact is used as input in a 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.

6.7.4 Performance Script

A 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 automatically or manually. In the former, scripts are generated through tools that use capture and replay mechanisms. On the latter one, scripts are generated through a programming language code.

6.7.5 Workload

This artifact is responsible for modifying the SUT situation through its different configurations. A workload may vary based on the test approach and it includes many 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.

6.7.6 Performance Scenario

In performance testing, the SUT must deal with different usage conditions. Those conditions mapped out with their particularities are defined here as test scenarios. A scenario maps out a given application context, within a determinate workload for a user profile. It should be modeled on basis of usage patterns and log files. In other words, a scenario must reflect real or expected system usage for performance testing.

6.7.7 Test Data

Test data consists of the test inputs. As in the performance test, one of the main objectives is the detection of application bottlenecks, the test data should be as close as possible to the application's production environment. Therefore, the results obtained with test data application are as reliable as possible.

6.7.8 Performance Testing Report

Test execution should produce reporting data. A technical report must contain test results, organized in a way that allows their interpretation by stakeholders.

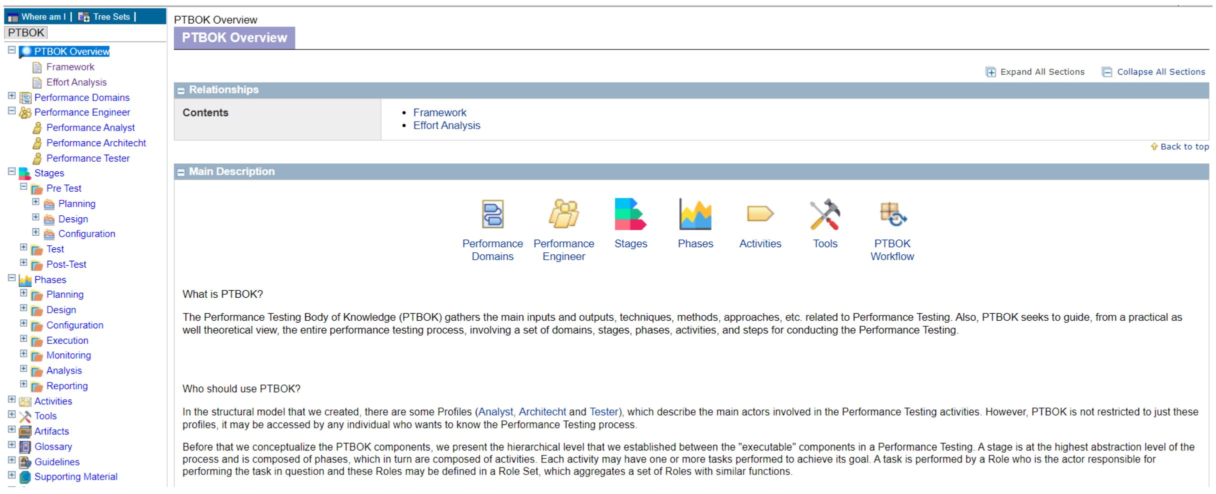
6.8 Guidelines

We have separated the various components present in the test activities into guidelines. Each activity may be supported with one or more of these components, and each guideline, in turn, can be employed in one or more activities. In the modeling of PTBOK, we have separated the guidelines by phases and it is also possible to view them grouped in a single set. As an example of a guideline, we can mention the list of tools, which details information about tools for performance testing. Other examples of guidelines are the different approaches that can be used in the test, such as load, stress, spike, among others.

6.9 Process Modeling

To model the process we use SPEM, with the EPF-Composer[1] support. We instantiate the main components of the metamodel to model the PTBOK[2]. Figure 34 shows a screenshot of the modeled process. On the left side menu, one can see the process views. In the center of the figure, we can see the process workflow. This workflow presents some tabs, in which it is possible to view some information such as the work breakdown structure, team allocation, and work product usage.

Figure 34 – PTBOK Process Modeling



Contents	• Framework • Effort Analysis

What is PTBOK?

The Performance Testing Body of Knowledge (PTBOK) gathers the main inputs and outputs, techniques, methods, approaches, etc. related to Performance Testing. Also, PTBOK seeks to guide, from a practical as well theoretical view, the entire performance testing process, involving a set of domains, stages, phases, activities, and steps for conducting the Performance Testing.

Who should use PTBOK?

In the structural model that we created, there are some Profiles (Analyst, Architecht and Tester), which describe the main actors involved in the Performance Testing activities. However, PTBOK is not restricted to just these profiles, it may be accessed by any individual who wants to know the Performance Testing process.

Before that we conceptualize the PTBOK components, we present the hierarchical level that we established between the "executable" components in a Performance Testing. A stage is at the highest abstraction level of the process and is composed of phases, which in turn are composed of activities. Each activity may have one or more tasks performed to achieve its goal. A task is performed by a Role who is the actor responsible for performing the task in question and these Roles may be defined in a Role Set, which aggregates a set of Roles with similar functions.

Source – Author.

We separate the process into views, which provide a general and detailed visualization. Also, the use of this metamodel provided a visualization with navigability between the components and keeps the artifacts and parts of the process gathered and organized in a central repository.

[1] Available at:<https://www.eclipse.org/epf/downloads/tool/tool_downloads.php>
[2] PTBOK is available at<lesse.com.br/ptbok>

6.9.1 SPEM x PTBOK Mapping

To model our process, we need to map each element of the SPEM to its respective element in PTBOK. Tables 5.a and 5.b shows this relationship between the elements. We followed the SPEM nomenclature pattern with some changes. The Task Definition we define as an Activity. Disciplines represent the Performance Domains. For Stages and Phases, we created custom categories, and the Guidances represent the PTBOK Guidelines.

Table 5 – Mapping from the PTBOK to SPEM

(a) Method Content

SPEM	PTBOK
Work Product Definition	Work Product
Role Definition	Role
Task Definition	Activity
Category/Discipline	Performance Domain
Category/Role Set	Role Set
Category/Custom	Stages
Category/Custom	Phases
Guidances	Guidelines

(b) Process

SPEM	PTBOK
Work Product Use	Work Product Use
Role Use	Role Use
Task Use	Activity Use
Discipline	Performance Domain
Guidances	Guidelines

Regarding Process (Table 5.b), the PTBOK maintains the SPEM nomenclature, except for the Task Use that we map the PTBOK activities and the SPEM guidances representing the PTBOK guidelines.

6.10 Chapter Summary

In this chapter, we introduced PTBOK, detailing each of its components and how they contribute to performance testing. Finally, we presented how the process was modeled and the mapping between the components of the SPEM and those instantiated in PTBOK.

7 EVALUATION

In this chapter, we present the survey that we conducted to evaluate the PTBOK. Section 7.1 details the survey protocol. In Section 7.3, we present the main threats to the study and how we mitigate them. Section 7.2 provides a discussion about survey results. We intended to carry out another type of evaluation, a *quasi*-experiment. However, due to the pandemic, we had to resize the research design and adopt a questionnaire to evaluate the PTBOK.

7.1 Survey Protocol

In this evaluation, one more time, we follow the protocol proposed by Kasunic (2005), as well as the survey that we detail in Chapter 5.

The main objective of the survey is to evaluate the modeled process (PTBOK) from the point of view of experts in Performance Testing. This evaluation aims to get improvement suggestions and possible issues. Also, participants will be able to indicate the PTBOK applicability in the software industry.

Our target audience may encompass industry professionals who work directly or indirectly with performance testing, which characterize the relevant population for our study. These subjects may have different roles in their companies, the main relevance for being part of the research is the knowledge about the performance testing process in their organizations.

To obtain a relevant sample for the research, we adopted two (2) different strategies. The first one is through direct contact with IT companies as well as specific performance testing organizations. The second one is the survey dissemination in the social network (LinkedIn[1]). For the latter, we specifically were looking for professional profiles with a large experience in performance testing and who currently perform roles related to performance testing in their companies.

To develop the questionnaire, we applied the same practices that we described in Section 5.1.4. We used the Google Forms[2] tool for the elaboration and analysis of survey artifacts.

The survey was composed of the following artifacts:

- **Consent Form:** To record the participants' agreement to participate in the assessment;
- **Profile Questionnaire:** To collect information about the professional profile of the participants;
- **Presentation Video:** Video containing general leveling information about Performance Testing and the PTBOK presentation;

[1] LinkedIn:<https://www.linkedin.com>
[2] Google Forms:<https://docs.google.com/forms/>

- **Technical Questionnaire:** Main form, containing the questions relevant to the evaluation of PTBOK. We divided this questionnaire into 5 Sections.

  In Section 1, we collect some personal data from participants such as names and emails.

  In Section 2, we discussed the PTBOK framework. So, we present the PTBOK framework overview in Figure 15 and then some questions related to the completeness and suitability of the process. Besides, we collected information about improvements and corrections suggested to PTBOK. In this section we use open and closed questions, aiming to increase the quantity and quality of the information in the answers.

  Section 3 discussed Figure 16, which refers to the effort analysis in the phases and stages of PTBOK.

- **Perceived Usefulness (PU):** Form containing 5 questions relevant to the evaluation of PTBOK perceived usefulness (Section 4);

- **Ease of Use (EoU):** Form containing 5 questions relevant to the evaluation of PTBOK ease of use (Section 5).

  The reference model used to formulate perceived usefulness and ease of use questions was the TAM (Technology Text Acceptance) Model, proposed by Davis (DAVIS, 1993).

Before applying the survey artifacts, we conducted a pilot with a similar sample to our target audience. This pilot enabled us to validate and make improvements to the survey protocol and artifacts. There were no significant changes in the artifacts from the pilot we ran, just a few minor grammatical adjustments.

We adopted two strategies for distributing the questionnaire. The first one was through a network of professionals from companies that apply performance testing. The second strategy was to search on LinkedIn[3], profiles involved in performance testing roles. When contacting the participants, we detailed some information such as the confidentiality of the data and the duration of the evaluation.

7.2 Results

The survey was available from December 2020 to May 2020[4]. Eight (8) participants performed the assessment, all of whom declared to have experience with performance testing, having worked in organizations that apply performance testing, such as Dell[5],

[3] LinkedIn:<https://www.linkedin.com>
[4] Survey data are available in:<https://bit.ly/3fAIgsH>
[5] Dell:<www.dell.com>

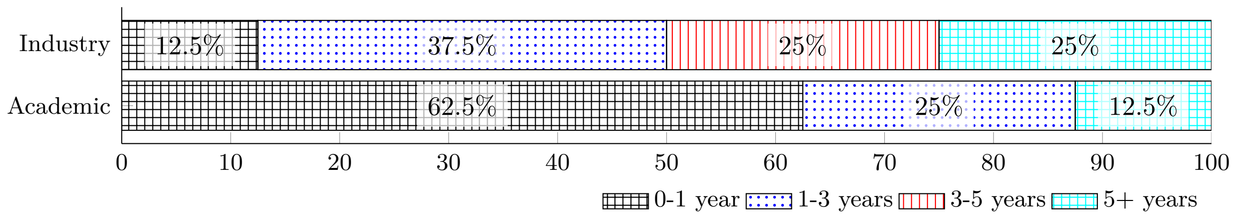
IBM[6], and Netflix[7]. Besides, one of the participants also reported having experience in this field in academia.

- **Profile Questionnaire**

Survey participants have the following profile. Regarding gender, there are two women (25%) and six men (75%). Three participants work as Software Architects (37,5%), and the others occupy the positions of University Professor, Performance Testing Engineer, Developer, Performance Tester, and Performance Engineer, 12,5% each.

Figure 35 presents data related to the subjects' experiences. The graph shows results relating to the experience in the academic field, 62.5% have experience from 0 to 1 year, 25% from 1 to 3 years, and 12.5% have more than five (5) years experience in this area. On the other hand, the experience in the industry area presents other data. 12.5% have experience from 0 to 1 year, 37.5% have experience from 1 to 3 years, 25% from 3 to 5 years, and 25% reported more experience than five (5) years in the field.

Figure 35 – Subjects experience.



Source – Author.

- **Technical Questionnaire (TQ)**

Regarding the **Technical Questionnaire (TQ)**, we analyzed the responses from two perspectives, internal cohesion of responses by applying the Cronbach's Alpha analysis, and individual response analysis. Cronbach's Alpha to TQ, which resulted in $\alpha = 0.857$, *i.e.* respondents tend to answer the same way for all TQ questions (TQ.1 - TQ.8). Figure 36 summarizes the results regarding each TQ question.

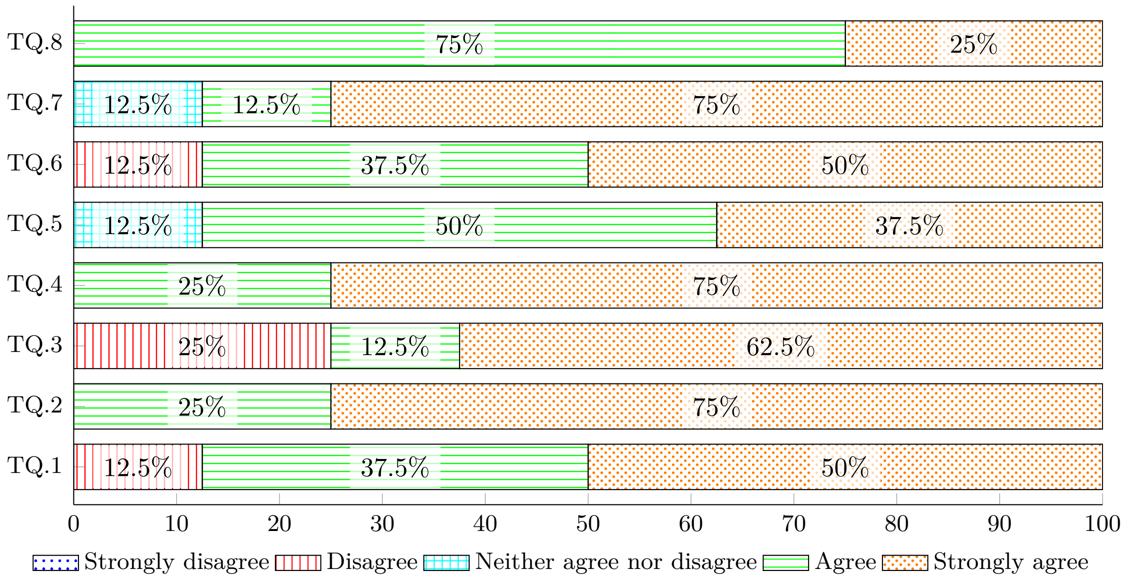
In TQ.1, we addressed the adequacy of the division among the stages of the PTBOK We can notice that 87.5% of the participants partially or totally agree that the division is adequate. For 75% of the participants, the PTBOK performance domains and activities are adequate (TQ.2 and TQ.4), and for 62.5% of the participants, the PTBOK phases are adequate (TQ.3). 87.5% of the participants think that the flow of the performance test process shown in PTBOK is correct and can assist in the performance testing

[6] IBM:<www.ibm.com>
[7] Netflix:<www.netflix.com>

of web applications (TQ.5). Also, this same percentage of participants agrees that it is possible to use the PTBOK in performance testing teams with segregation of responsibilities (TQ.6). Regarding the use of PTBOK to support teaching in academia, 75% agree that PTBOK may assist in teaching activities in the performance testing area (TQ.7). In the last objective question of the technical questionnaire (TQ.8), we present the figure that shows the PTBOK estimated effort.

100% of the participants agreed that the figure adequately represents the effort employed in PTBOK performance domains, phases, and stages.

Figure 36 – Technical Questionnaire



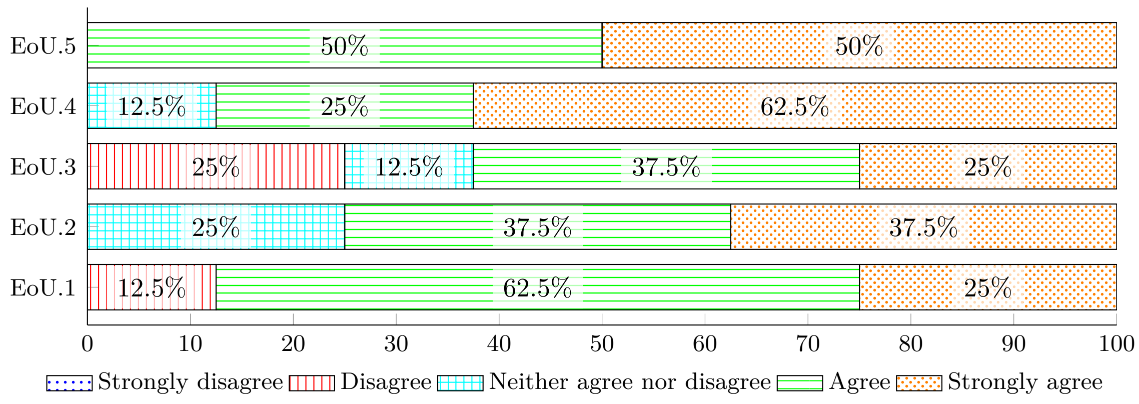
Source – Author.

- **Ease of Use**

Regarding the **Ease of Use (EoU)**, as well as TQ, we analyzed the responses from two perspectives, internal cohesion of responses by applying the Cronbach's Alpha analysis, and individual response analysis. Cronbach's Alpha to EoU, which resulted in $\alpha = 0.887$, *i.e.* respondents tends to answer the same way for all EoU questions (EoU.1 - EoU.5).

Figure 37 summarizes the results regarding each EoU question. In question EoU.1 we stated the PTBOK ease of use. The wide majority answered positively to this question, 87.5% answered *strongly agree* or *agree*. 75% of the participants will recommend the use of PTBOK (EoU.2). 87.5 % of the participants agree that PTBOK is a good idea and that it has easy access (EoU.3 and EoU.4). In the EoU.5 question, 50% *agree* that PTBOK can facilitate the performance testing, while the remaining 50% answered *strongly agree* to EoU.5 affirmation.

Figure 37 – Ease of Use



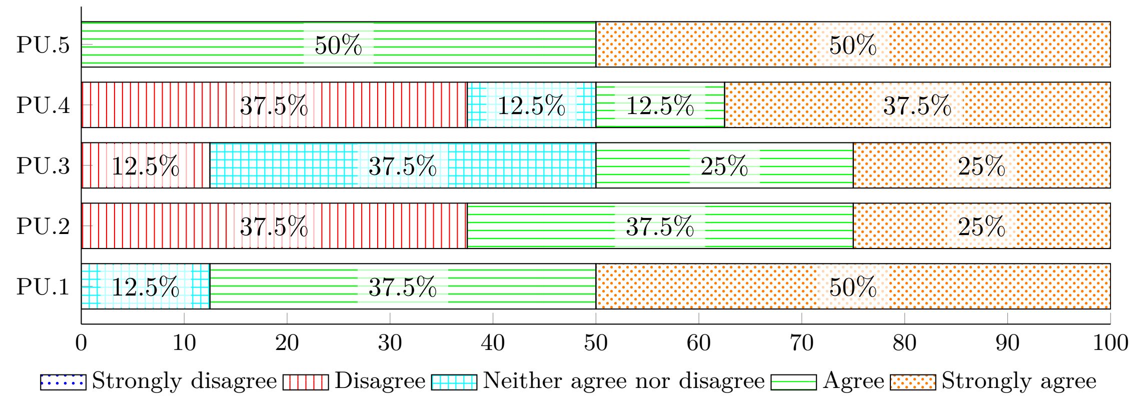
Source – Author.

- **Perceived Usefulness**

Regarding the **Perceived Usefulness (PU)**, as well as TQ and EoU, we analyzed the responses from two perspectives, internal cohesion of responses by applying the Cronbach's Alpha analysis, and individual response analysis. Cronbach's Alpha to PU, which resulted in $\alpha = 0.887$, *i.e.* respondents tend to answer the same way for all PU questions (PU.1 - PU.5).

Figure 38 summarizes the results regarding each PU question. 87.5% agree that PTBOK is useful for performance testing conduction (PU.1). In the PU.2 question, although 62.5% agree that PTBOK can increase production in conducting the test, a significant number (37.5%) partially disagree with this statement. 50% of the participants agree that PTBOK produces results expected for a performance test process and that it can decrease the time spent in test conduction (PU.3 and PU.4). Finally, in the PU.5 question, all the participants enjoyed using PTBOK, with 50% agreeing and 50% strongly agreeing.

Figure 38 – Perceived Usefulness



Source – Author.

- **Open Questions**

We elaborated nine (9) open questions related to technical aspects of PTBOK. To extract relevant considerations in open questions, we use the Dedoose[8] tool for coding support. After analyzing the responses, we identified 3 (three) main codes: (I) Improvement Suggestions, (II) Applicability, and (III) Positive Aspects, which we detail below.

I - Improvement Suggestions

This code concerns the improvements in PTBOK pointed out by the participants, which we discuss below.

R1 suggested the inclusion of some activities that have a warm-up, listing some tools that support it. For R4, it may be necessary to include an iteration path and the leading of alternative flows after the test execution. In addition, R4 and R6 also suggested further details of the roles involved in the process.

R7 presented some suggestions related to the order of some activities, such as the choice of the test method before the tool choice. Another aspect addressed by R7 was that, in the analysis phase, it would possibly be more appropriate to build graphs and reports after evaluating the SLA and requirements, through analysis with previously published data, which could facilitate understanding.

II - Applicability

This code is related to the most suitable applicability to PTBOK, mainly under two contexts: an organizational one, involving the size of teams and organizations, and another identifying which software development methodologies would be most appropriate for the use of PTBOK.

R1, R6, and R7 believe that PTBOK can be applied both to teams that work with more traditional methodologies and to teams that adopt agile methods. In addition, R7 justifies it by mentioning the ease of understanding the process as a facilitator of its application. However, for R3 there is a lot of overhead, which may make it difficult to apply it in agile processes.

Regarding the size of the organizations in which the application of PTBOK would be more appropriate, in general, the participants answered that medium to large companies would have better applicability since there is a greater probability of having dedicated performance testing teams. However, R5 understood the main concept when we conceived PTBOK, which is its adaptability to different organizations, with the responsibility of tailoring the process according to their organizational structure and particularities. R2 states that because it does not perceive feedback loop/back propagation signaling, it looks like a process for waterfall teams. Using Agile, it would be necessary to merge part of the activities and make the points of iterative/refinement evident.

III - Positive Aspects

[8] Dedoose:<https://www.dedoose.com/>

This code addresses the most relevant aspects pointed out by the participants. We seek to understand the main strengths of PTBOK from the professional opinion.

R1 succinctly defines his opinion of PTBOK with "***It looks great!***". For R2 PTBOK is a very valuable effort in trying to standardize something that, in practice, is often done without adequate preparation. R3 expressed that this process can be of great support due to the clarity of the tasks and organization. The defined granularity was mentioned by R4 as a positive point, emphasizing also that PTBOK contemplates the main aspects of the performance testing. R5 says "***I think it is very well documented and has a great visual and navigation quality. The process is easy to follow and looks very instructive.***" For R6, PTBOK represents an organized and well-structured way of describing the performance process. It brings clarity and a wide understanding that is often lacking to those who execute the process and to the stakeholders of a performance team.

7.3 Threats to Validity

Construct Validity: Some threats may affect the validity of the constructor and are related to the possibility of its generalization. To mitigate threats related to the construct validity, we validated all artifacts produced by a performance testing expert, also conducted a pilot questionnaire to gather issues and improvements to survey artifacts. However, we were unable to mitigate the low statistical power related to the number of participants.

Internal Validity: Threats to internal validity are forces that may alter the independent variable. For the elaboration of the questionnaire, the questions were grouped by similarity, and different strategies were used (open and closed questions). Also, we were careful in synthesizing issues to avoid participants' fatigue as much as possible.

External Validity: External threats can limit our ability to generalize the results of the experiment externally. To mitigate threats related to the sample size, so that we could find a meaningful sample for our context, different means of participants prospecting were applied. We contacted participants through a social network and by direct contact with IT companies.

Conclusion Validity: This type of threat is related to issues that may affect the correct conclusions based on the relationship between treatments and the results of the experiment. To mitigate these threats, we applied some strategies such as analysis using tool-assisted coding and Cronbach's Alpha analysis.

7.4 Chapter Summary

In this chapter, we presented the PTBOK evaluation. We conducted a survey in which we collected the opinion of professionals in the performance testing area. By means of this evaluation, it was possible to gather info that allows us to understand positives aspects, applicability, and improvements for the PTBOK. In this survey, we changed our adherence strategy a little. Despite having a smaller number of participants, compared to the survey described above, we obtained detailed responses and representative feedback.

8 CONCLUSION

Performance testing can bring numerous benefits to companies. With performance testing, they may manage scalability, assess the adequacy of developed software performance, and improve the performance. A defined process may assist those who deal with performance testing in conducting this type of test (MEIER et al., 2007).

Firstly, an SLR gave us an academic perspective on performance testing. This revision provides a feature model that was an initial basis for PTBOK. Then, through a survey, we sought to understand the industry's position concerning performance testing. Therefore, we merged SLR to survey the results so that we could start the conception of PTBOK. Hence, we mapped concepts of the SPEM metamodel and instantiated them on PTBOK. So, we finally evaluated our study, through a survey with experts in the performance testing area.

Our main contributions are listed below:

1. An SLR collecting, analyzing, and discussing thirty-seven (37) different works on performance testing area;
2. A Feature Model with the main concepts related to performance testing;
3. A Survey with relevant information to the performance testing process from an industry perspective;
4. Modeling of PTBOK in a concise metamodel.

We list some publications below that were originated in the context of our research.

Published:

- Norberto, M., Gaedicke, L., Bernardino, M., Legramante, G., Basso, F. P., & Rodrigues, E. M. (2019, October). Performance Testing in Mobile Application: a Systematic Literature Map. In *Proceedings of the XVIII Brazilian Symposium on Software Quality* (pp. 99-108).
- Girardon, G., Costa, V., Machado, R., Bernardino, M., Legramante, G., Basso, F. P., & Neto, A. (2020, March). Testing as a service (TaaS) a systematic literature map. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing* (pp. 1989-1996).
- Costa, V., Girardon, G., Bernardino, M., Machado, R., Legramante, G., Neto, A., & de Macedo Rodrigues, E. (2020, March). Taxonomy of performance testing tools: a systematic literature review. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing* (pp. 1997-2004).
- Legramante, Guilherme, et al. "Systematic Literature Review on Web Performance Testing." Anais da IV Escola Regional de Engenharia de Software. SBC, 2020.

Under Review:

- Towards a Hybrid Process Modeling Approach based on SPEM and BPMN: the Performance Testing Body of Knowledge. Submitted to Empirical Software Engineering and Measurement - ESEM (Main Track).

Future Submission:

- Towards a Performance Testing Body of Knowledge (PTBOK). Future submission to a Journal.

Open Questions

After conducting our study, some questions remain open. Although PTBOK allows adaptability to traditional and agile software models, evidence obtained in the open questions show that in the current form, PTBOK is more suitable for traditional methods such as waterfall or RUP and for teams that have well-defined roles in performance testing conduction. Therefore, a relevant issue is an adaptation in the process to make this more practical for agile teams so that a professional can act in different roles and use methodologies such as SCRUM and XP.

We provide guidelines that may assist in performance testing conduction. However, there is not a case study directed to the industry yet, which applies these guidelines to teams that do not have experience with performance testing, or even that want to start applying this type of test in their organizations.

Another relevant issue is the possibility of including a phase or activities related to the testing warm-up. An applicability study is needed to verify how to merge this warm-up to PTBOK.

Finally, we also give as an open question, the possibility of modeling the process with some other technology or meta-model, or even the application of mechanisms that allow better usability to the current PTBOK version.

Future Work

As future work, we will continue to investigate inputs to increase the body of knowledge maintained at PTBOK. In addition, we are working on the design of a management tool for performance testing. Through it, we intend to use PTBOK as support and to coordinate the technologies used in the test, keeping performance testing projects in a repository that can assist organizations in managing performance testing, while also maintaining the historical data on tests performed.

REFERENCES

A Guide to the Business Analysis Body of Knowledge (BABOK guide). [S.l.: s.n.]. Cited at page 25.

ABRAN, A. et al. Software engineering body of knowledge. **IEEE Computer Society, Angela Burgess**, 2004. Cited at page 25.

ALI, A.; BADR, N. Performance testing as a service for web applications. In: **2015 IEEE 7th International Conference on Intelligent Computing and Information Systems, ICICIS 2015**. Cairo, Egypt: [s.n.], 2015. p. 356–361. Cited 2 times at pages 33 and 39.

ANDERSON, K. S. et al. SWORD: Scalable and Flexible Workload Generator for Distributed Data Processing Systems. In: **Proceedings of the 38th Conference on Winter Simulation**. [S.l.]: Winter Simulation Conference, 2006. (WSC '06), p. 2109–2116. Cited 2 times at pages 33 and 39.

ARORA, J. Web testing using UML environment models. In: **2016 International Conference on Computing, Communication and Automation (ICCCA)**. [S.l.: s.n.], 2016. p. 785–789. Cited 2 times at pages 33 and 39.

BERNARDINO, M.; ZORZO, A. F.; RODRIGUES, E. M. Canopus: A domain-specific language for modeling performance testing. In: IEEE. **2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)**. [S.l.], 2016. p. 157–167. Cited 4 times at pages 17, 23, 33, and 39.

BLACK, R.; ROMMENS, J. L.; AALST, L. V. D. **The Expert Test Manager: Guide to the ISTQB Expert Level Certification**. [S.l.]: Rocky Nook, Inc., 2017. Cited at page 18.

BOONE, B. et al. SALSA: QoS-aware load balancing for autonomous service brokering. **Journal of Systems and software**, v. 83, n. 3, p. 446–456, 2010. Cited 2 times at pages 33 and 39.

BOURQUE, P. et al. The guide to the software engineering body of knowledge. **IEEE software**, IEEE, v. 16, n. 6, p. 35–44, 1999. Cited 2 times at pages 24 and 25.

BRAGA, R. et al. A Machine Learning Approach to Generate Test Oracles. In: **Proceedings of the XXXII Brazilian Symposium on Software Engineering**. New York, NY, USA: ACM, 2018. (SBES '18), p. 142–151. Cited 2 times at pages 33 and 39.

CAMARGO, A. A. de et al. An Architecture to Automate Performance Tests on Microservices. In: **Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services**. [S.l.]: ACM, 2016. Cited 2 times at pages 33 and 39.

CHEN, S. et al. Towards Practical Modeling of Web Applications and Generating Tests. In: **2010 4th IEEE International Symposium on Theoretical Aspects of Software Engineering**. [S.l.: s.n.], 2010. p. 209–217. Cited 2 times at pages 33 and 39.

DALAL, S. R. et al. Model-based testing in practice. In: **Proceedings of the 21st international conference on Software engineering**. [S.l.: s.n.], 1999. p. 285–294. Cited at page 17.

DAVIS, F. D. User acceptance of information technology: system characteristics, user perceptions and behavioral impacts. v. 38, n. 3, p. 475 – 487, 1993. Cited at page 78.

ELVESAETER, B.; BENGURIA, G.; ILIEVA, S. A comparison of the essence 1.0 and spem 2.0 specifications for software engineering methods. In: **Proceedings of the Third Workshop on Process-Based Approaches for Model-Driven Engineering**. [S.l.: s.n.], 2013. p. 1–10. Cited 2 times at pages 9 and 26.

FERRARI, D. On the foundations of artificial workload design. In: **Proceedings of the 1984 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems**. New York, NY, USA: ACM, 1984. (SIGMETRICS '84), p. 8–14. Cited at page 24.

FREITAS, A.; VIEIRA, R. An ontology for guiding performance testing. In: IEEE. **2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)**. [S.l.], 2014. v. 1, p. 400–407. Cited 4 times at pages 23, 33, 36, and 39.

GAO, H.; LI, Y. Generating quantitative test cases for probabilistic timed Web Service Composition. In: **Proceedings - 2011 IEEE Asia-Pacific Services Computing Conference, APSCC 2011**. [S.l.: s.n.], 2011. p. 275–283. Cited 2 times at pages 33 and 39.

GARG, N.; SINGLA, S.; JANGRA, S. Challenges and Techniques for Testing of Big Data. **Procedia Computer Science**, v. 85, p. 940–948, 2016. Cited 2 times at pages 33 and 39.

GIAS, A. U. et al. IVRIDIO: Design of a software testing framework to provide Test-first Performance as a service. In: **Third International Conference on Innovative Computing Technology (INTECH 2013)**. [S.l.: s.n.], 2013. p. 520–525. Cited 2 times at pages 33 and 39.

HADHARAN, R. et al. End to End Performance Modeling of Web Server Architectures. **SIGMETRICS Perform. Eval. Rev.**, ACM, New York, NY, USA, v. 28, n. 2, p. 57–63, 2000. Cited 2 times at pages 33 and 39.

HANMER, R. S.; LETOURNEAU, J. P. A best practice for performance engineering. **Bell Labs technical journal**, Nokia Bell Labs, v. 8, n. 3, p. 75–89, 2003. Cited 2 times at pages 33 and 39.

HUANG, X. et al. An adaptive performance modeling approach to performance profiling of multi-service web applications. In: **Proceedings - International Computer Software and Applications Conference**. [S.l.: s.n.], 2011. p. 4–13. Cited 3 times at pages 33, 36, and 39.

JURIC, M. B. et al. Comparison of performance of Web services, WS-Security, RMI, and RMI–SSL. **Journal of Systems and Software**, v. 79, n. 5, p. 689–700, 2006. Cited 2 times at pages 33 and 39.

KASUNIC, M. **Designing an effective survey**. [S.l.], 2005. Cited 3 times at pages 9, 43, and 77.

KIM, G.-H.; KIM, Y.-G.; CHUNG, K.-Y. Towards virtualized and automated software performance test architecture. **Multimedia Tools and Applications**, Springer, v. 74, n. 20, p. 8745–8759, 2015. Cited 2 times at pages 9 and 24.

KITCHENHAM, B. A. **Guidelines for performing Systematic Literature Reviews in software engineering. EBSE Technical Report EBSE-2007-01**. [S.l.: s.n.], 2007. Cited at page 27.

KOZIOLEK, H. Goal, question, metric. In: **Dependability Metrics**. [S.l.]: Springer, 2008. p. 39–42. Cited 3 times at pages 11, 27, and 31.

KUN, W. et al. Performance analysis of the OGSA-DAI 3.0 software. In: **Proceedings - International Conference on Information Technology: New Generations, ITNG 2008**. [S.l.: s.n.], 2008. p. 15–20. Cited 2 times at pages 33 and 39.

LAUDAN, L. **Science and hypothesis: Historical essays on scientific methodology**. [S.l.]: Taylor & Francis, 1981. v. 19. Cited at page 21.

LIU, X. et al. Distributed Testing System for Web Service Based on Crowdsourcing. **Complexity**, v. 2018, 2018. Cited 2 times at pages 33 and 39.

MARCHEZAN, L. et al. Thoth: A web-based tool to support systematic reviews. In: IEEE. **2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)**. [S.l.], 2019. p. 1–6. Cited at page 27.

MARSZALKOWSKI, J. Prototype of high performance scalable advertising server with local memory storage and centralised processing. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. Budapest, Hungary: [s.n.], 2012. p. 194–203. Cited 2 times at pages 33 and 39.

MEIER, J. et al. **Performance testing guidance for web applications: patterns & practices**. [S.l.]: Microsoft press, 2007. Cited 7 times at pages 17, 23, 33, 36, 39, 49, and 85.

MEMON, A. M.; SOFFA, M. L. Regression testing of GUIs. **ACM SIGSOFT Software Engineering Notes**, ACM, v. 28, n. 5, p. 118–127, 2003. Cited at page 17.

MIRSHOKRAIE, S.; MESBAH, A.; PATTABIRAMAN, K. Guided Mutation Testing for JavaScript Web Applications. **IEEE Transactions on Software Engineering**, v. 41, n. 5, p. 429–444, 2015. Cited 2 times at pages 33 and 39.

MOLYNEAUX, I. **The art of application performance testing: Help for programmers and quality assurance**. [S.l.]: O'Reilly Media, Inc., 2009. Cited at page 23.

OMG. **Software Process Engineering Metamodel SPEM 2.0 OMG Draft Adopted Specification**. [S.l.], 2006. Cited 3 times at pages 18, 25, and 26.

OMG, O.; PARIDA, R.; MAHAPATRA, S. Business process model and notation BPMN version 2.0. **Object Management Group**, v. 1, n. 4, 2011. Cited at page 18.

PFAU, J.; SMEDDINCK, J. D.; MALAKA, R. 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**. New York, NY, USA: ACM, 2017. (CHI PLAY '17 Extended Abstracts), p. 153–164. Cited 4 times at pages 33, 35, 36, and 39.

PMI. **A guide to the project management body of knowledge (PMBOK guide)**. [S.l.]: Project Management Inst, 2000. v. 2. Cited at page 25.

PONS, A. P. Improving the performance of client Web object retrieval. **Journal of Systems and Software**, v. 74, n. 3, p. 303–311, 2005. Cited 2 times at pages 33 and 39.

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição**. [S.l.]: Editora Feevale, 2013. Cited at page 21.

PUTRI, M. A.; HADI, H. N.; RAMDANI, F. Performance testing analysis on web application: Study case student admission web system. In: IEEE. **2017 international conference on sustainable information engineering and technology (SIET)**. [S.l.], 2017. p. 1–5. Cited 2 times at pages 33 and 39.

RODRIGUES, E. et al. PLeTsPerf a model-based performance testing tool. In: IEEE. **2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)**. [S.l.], 2015. p. 1–8. Cited 3 times at pages 33, 36, and 39.

RODRIGUES, E. M. et al. Evaluating Capture and Replay and Model-based Performance Testing Tools: An Empirical Comparison. In: **Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement**. [S.l.]: ACM, 2014. Cited 2 times at pages 33 and 39.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Malaysia; Pearson Education Limited,, 2016. Cited at page 31.

SHARIFI, M.; TASHARROFI, S.; MAHMOUDZADEH, H. A new method on automated web application testing. **WSEAS Transactions on Computers**, v. 4, n. 11, p. 1684–1691, 2005. Cited 2 times at pages 33 and 39.

SHULL, F.; SINGER, J.; SJØBERG, D. I. **Guide to advanced empirical software engineering**. [S.l.]: Springer, 2007. Cited at page 44.

SNODGRASS, R. A relational approach to monitoring complex systems. **ACM Transactions on Computer Systems (TOCS)**, ACM, v. 6, n. 2, p. 157–195, 1988. Cited 2 times at pages 33 and 39.

SOUZA, F. C. et al. **Automating Search Strings for Secondary Studies**. [S.l.]: Springer International Publishing, 2018. 839–848 p. Cited at page 31.

SOUZA, T. Silva-de; TRAVASSOS, G. H. Observing Effort Factors in the Test Design & Implementation Process of Web Services Projects. In: **Proceedings of the 2Nd Brazilian Symposium on Systematic and Automated Software Testing**. New York, NY, USA: ACM, 2017. (SAST), p. 7:1—-7:10. Cited 2 times at pages 33 and 39.

SPRENKLE, S. et al. Automated Replay and Failure Detection for Web Applications. In: **Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering**. New York, NY, USA: ACM, 2005. (ASE '05), p. 253–262. Cited 2 times at pages 33 and 39.

STER, D. C. van der et al. Hammercloud: A stress testing system for distributed analysis. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2011. v. 331, n. 7, p. 072036. Cited 3 times at pages 33, 35, and 39.

SUBMITTERS, O. Essence–kernel and language for software engineering methods. Citeseer, 2012. Cited at page 18.

SUBRAYA, B. M. **Integrated approach to web performance testing: A practitioner's guide**. [S.l.: s.n.], 2006. 1–368 p. Cited 7 times at pages 17, 18, 33, 35, 36, 37, and 39.

TSELIKIS, C.; MITROPOULOS, S.; DOULIGERIS, C. An evaluation of the middleware's impact on the performance of object oriented distributed systems. **Journal of Systems and Software**, v. 80, n. 7, p. 1169–1181, 2007. Cited 3 times at pages 33, 35, and 39.

WOHLIN, C. et al. **Experimentation in software engineering**. [S.l.]: Springer Science & Business Media, 2012. Cited 2 times at pages 27 and 47.

WOODSIDE, M.; FRANKS, G.; PETRIU, D. C. The future of software performance engineering. In: **2007 Future of Software Engineering**. [S.l.: s.n.], 2007. p. 171–187. Cited 2 times at pages 17 and 23.

XIA, J. et al. An empirical performance study on PSIM. **Computer Journal**, v. 49, n. 5, p. 509–526, 2006. Cited 2 times at pages 33 and 39.

XIA, X. et al. Multi-level logs based web performance evaluation and analysis. In: **ICCASM 2010 - 2010 International Conference on Computer Application and System Modeling, Proceedings**. [S.l.: s.n.], 2010. v. 4, p. V437–V441. Cited 2 times at pages 33 and 39.

XU, X. et al. URMG: Enhanced CBMG-based method for automatically testing web applications in the cloud. **Tsinghua Science and Technology**, v. 19, n. 1, p. 65–75, 2014. Cited 3 times at pages 33, 35, and 39.

YIN, J. et al. 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**. Zhang Jia Jie, Hunan, China: [s.n.], 2008. p. 1285–1291. Cited 3 times at pages 33, 36, and 39.

INDEX