Availability Evaluation of a Learning Management Environment

Gervasio Eufrauzino Teixeira
get@cin.ufpe.br
Federal University of Pernambuco
Recife, Pernambuco, BR

Erick Barros Nascimento
ebn@cin.ufpe.br
Federal University of Pernambuco
Recife, Pernambuco, BR

Joao Ferreira da Silva Junior
jfjunior@dei.uc.pt
Universtity of Coimbra
Coimbra, PT

Paulo Romero Martins Maciel
prmm@cin.ufpe.br
Federal University of Pernambuco
Recife, Pernambuco, BR

Jamilson Ramalho Dantas
jrd@cin.ufpe.br
Federal University of Pernambuco
Recife, Pernambuco, BR

Abstract

Availability analysis is an essential aspect of evaluating a system's response to different demands. State models and combinatorial models provide a framework for quantitatively assessing availability, helping to identify hot spots and implement risk mitigation strategies. This work proposes developing and validating state space models and combinatorial models to analyze the availability of a Moodle installation in a LAMP stack. Validation of these models allows the construction of study scenarios to improve system uptime and user experience. With the creation of models in Reliability Block Diagrams (RBD) and Continuous-Time Markov Chains (CTMC), it was possible to validate a proposed scenario, achieving an availability of 99.65% with a confidence level of 95%.

Keywords

Availability, RBD, CTMC, LAMP, Moodle

Reference Format:
Gervasio Eufrauzino Teixeira, Erick Barros Nascimento, Joao Ferreira da Silva Junior, Paulo Romero Martins Maciel, and Jamilson Ramalho Dantas. 2024. Availability Evaluation of a Learning Management Environment. In *Proceedings of LADC '24: 13th Latin–American Symposium on Dependable and Secure Computing (1st Workshop on Resilience Engineering in Computer Systems).* , 6 pages.

1 Introduction

The rapid advancement of technology and challenges from the COVID-19 pandemic have driven the growth of distance learning platforms. Learning Management Systems (LMS) integrate traditional teaching with digital resources, providing personalized learning experiences [7]. Moodle stands out as the most popular open-source LMS, supported by a wide user base and easy deployment on a LAMP stack (Linux, Apache, MySQL, PHP) [1, 2, 15]. Reliability, availability, and resilience are critical for maintaining a high-quality learning environment. Availability analysis helps identify system weaknesses and improve resilience, using state models to assess system performance and mitigate risks quantitatively [12]. The complexity of LMSs like Moodle requires combinatorial models to evaluate component interactions and failure scenarios. Some work has been proposed for the availability evaluation of different kinds of systems. Jogi and Sinha [9] evaluated the performance of databases, including MySQL, in write-intensive operations. The study presented an application that used Tomcat and compared the performance of databases in terms of Transactions Per Second (TPS). The aim was to analyze the performance of various database systems. Wannapiroon et al. [17], in turn, developed a Cloud Learning Management System (CLMS) for Higher Education Institutions. The system was divided into modules and evaluated as excellent in efficiency and user satisfaction, with high scores on functional and security tests. Its tests include functional requirements testing, functional testing, usability testing, security testing, and performance, but no availability tests were carried out. The authors of the study [11] analyzed the interaction between PHP and MySQL in a Digital Asset Management System. The system was designed to ensure data security and integrity, using PHP for server logic and MySQL as the database. The implementation included security measures to protect stored data. Different from previous works, our work presents a methodology for evaluating the availability of Moodle, a topic not covered in depth by other articles. The methodology includes creating RBD and CTMC models and providing a detailed framework for availability analysis. Model validation through fault injection and repairs in specific system components (hardware, software, Apache, MySQL, PHP) guarantees the accuracy of the proposed models. This practical approach adds a layer of reliability that complements the more theoretical evaluations found in other studies.

The paper is organized as follows: Section 2 describes the proposed architecture and methodology. Section 3 details the availability models used in this analysis. Section 4 discusses a case study. Finally, Section 5 presents the conclusions and suggestions for future work.

2 Proposed Architecture and Methodology

In this section, we detail the proposed architecture and methodology for evaluating the availability of a Learning Management System (LMS) deployed on a LAMP stack.

2.1 Proposed Architecture

The architecture adopted for this study is presented in this subsection. Figure 1 illustrates the proposed architecture. It consists of two servers connected to the Internet through a switch, which provides access to the network and enables communication between the machines.

1st Workshop on Resilience Engineering in Computer Systems, Nov 26–29, 2024, Recife, BR

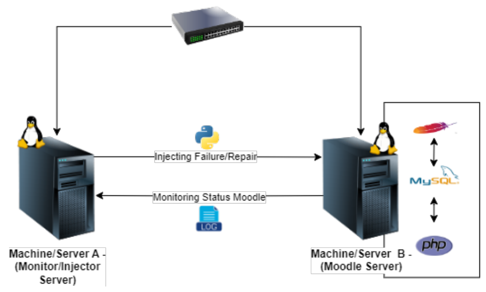

Injecting Failure/Repair
Monitoring Status Moodle
LOG
MySQL
php
Machine/Server A - (Monitor/Injector Server)
Machine/Server B - (Moodle Server)

Figure 1: Proposed architecture.

Machine A, a Linux 22.04 LTS server, runs fault injectors and repairs with exponentially distributed times across components of Machine B. The injectors are Python scripts, and Machine A also monitors the status (available or unavailable) of Moodle LMS hosted on Machine B. The fault injection scripts independently introduce failures and repairs across Machine B's hardware, OS, Apache, MySQL, and PHP, ensuring reliable execution of each process. Machine B, also a Linux 22.04 LTS server, hosts the LAMP stack and Moodle LMS version 4.3. Its availability depends on the continuous operation of critical components—hardware, OS, Apache, MySQL, and PHP. If any component fails, the system becomes temporarily unavailable. Machine A injects faults and repairs in each component of Machine B at exponentially distributed times and monitors Moodle's status every five seconds, logging timestamps with "UP" or "DOWN" based on component availability. This log data will later calculate Mean Time to Failure (MTTF) and Mean Time to Repair (MTTR) to assess system availability.

This configuration allows you to monitor how long Machine/Server B has been available or unavailable, which allows the application of the models to be created.

2.2 Methodology and Overview

The proposed methodology consists of seven steps, each designed to systematically evaluate the availability of the Learning Management System (LMS) deployed on a LAMP stack.

The first step involves defining the base scenario, which serves as a reference for the entire investigation. This base scenario includes all variables and initial conditions that will be considered throughout the study [3]. This step is crucial for ensuring the results are relevant and applicable to the study context [13].

The next step is configuring the environment where the experiments will be conducted. This includes preparing all the necessary resources, tools, and conditions. The server is instantiated with the LAMP stack in this phase. Proper environment configuration is vital for ensuring the validity and reliability of the data collected. Figure 2 presents the proposed methodology's flowchart.

After configuring the environment, the experimentation and data collection phase begins, where faults are injected into and repaired within the system components (hardware, operating system, Apache,

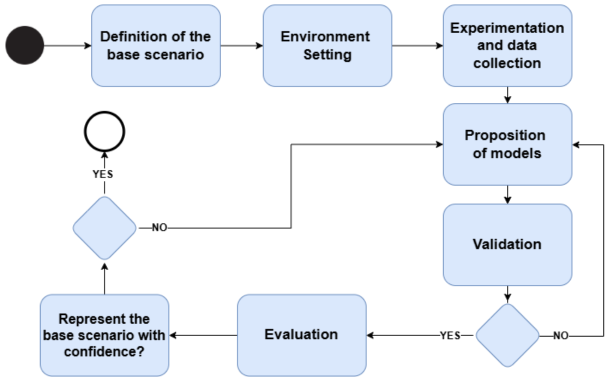
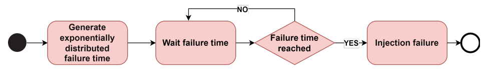
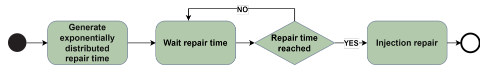

Definition of the base scenario
Environment Setting
Experimentation and data collection
Proposition of models
Validation
YES
NO
Evaluation
Represent the base scenario with confidence?
YES
NO

Figure 2: Proposed methodology.

MySQL, and PHP) according to the methodology illustrated in Figure 2. The times for fault injection and repair follow an exponential distribution to simulate conditions of unpredictability, as detailed further in Section 2.1. Machine A monitors the availability of the Moodle service every five seconds, recording this data in a log file that is subsequently used to calculate the Mean Time to Failure (MTTF) and Mean Time to Repair (MTTR), which are essential metrics for determining overall system availability. During this phase, a Python script running on Machine A generates exponentially distributed times for faults and repairs for the target component, checking if the failure time has been reached. If a failure occurs, the system enters a fault state; thereafter, a repair time is generated, and once reached, the component is restored, with a new failure time then set (see Figure 3).



(a) Failure process



(b) Repair process

Figure 3: Failure/Repair Injection Diagram.

After data collection, the next step is to propose models using Reliability Block Diagrams (RBD) and Continuous-Time Markov Chains (CTMC) to explain or predict observed phenomena. These models require validation to ensure accuracy, which involves comparing the results generated by the models with those from the base scenario. Only models that produce data within a 95% confidence interval proceed to further evaluation, while others are refined to improve alignment with the base scenario. In the evaluation stage, the validated models undergo a thorough analysis to confirm that they reliably represent the base scenario and are robust in the research context. This final stage consolidates the methodology, summarizing the main findings, and noting any limitations of the study, which serve as a basis for future research directions.

3 Availability Models

This section presents the two availability analysis models employed in the case study and defines the basic parameters for model validation.

3.1 RBD Model

Based on the architecture proposed in Section 2.1, an RBD representation of the system components was modeled. The model is based on five blocks in series, as follows:

- HW (Hardware): Refers to the physical equipment necessary to run the Operating System and services needed for Moodle;
- OS (Operating System): The base software that manages the hardware and offers essential services for the other components;
- Apache: The web server that handles HTTP requests and delivers content to users;
- PHP: A server-side scripting language used to run Moodle's dynamic pages;
- MySQL: A relational database management system that stores and manages Moodle data.

Reliability Block Diagrams (RBDs) are used to represent and analyze the availability of complex systems by decomposing the system into individual components, each represented as a block. The connections between these blocks indicate the functional dependencies among components. A series configuration is applied to systems where every component must function correctly for the system to operate. In such a configuration, the failure of a single component results in the failure of the entire system, reflecting the critical dependency on each component's functionality.

In this work, it is assumed that the system is available only if all components are active simultaneously (configured in a series arrangement). As a result, a sequential representation of the components is required, as illustrated by the RBD model in Figure 4.

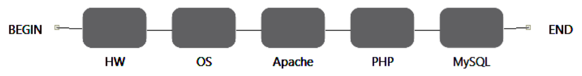


Figure 4: RBD – Moodle.

The serial configuration emphasizes the critical dependency of each component in the Moodle architecture. For example, if the hardware (HW) fails, the operating system (OS) cannot operate. If Apache is down, HTTP requests cannot be processed. If PHP fails, Moodle cannot execute its logic, and if MySQL fails, Moodle cannot access or store data. Thus, the integrity of each component is crucial to the system's full functionality.

3.2 CTMC Model

Continuous-time Markov Chains (CTMCs) model systems with stochastic behavior and state transitions over time [12]. In the CTMC model, Moodle is represented by states and exponentially distributed transition rates, allowing detailed analysis of dynamic processes such as component failures and repairs. This approach enables rigorous calculations of metrics like system availability.

CTMCs are ideal for systems where temporal dynamics and stochastic behavior are essential, such as maintenance and repair scenarios. They model failure events, recovery processes, and their durations, providing a detailed view of how these factors impact system availability. To facilitate reliability analysis, a CTMC model was derived from the Reliability Block Diagram (RBD) of Figure 4, in which the LAMP stack is represented as a single aggregated component. This model integrates the failure and repair parameters of the individual stack components. The component states are outlined in Table 1. The combination of these unified components is referred to as the APP.

Table 1: CTMC – Description of states

State	Description	System
UUU	Hardware, OS, APP available	Available
UUD	Hardware, OS available; APP unavailable	Unavailable
UDD	Hardware available; OS, APP unavailable	Unavailable
DDD	Hardware, OS, APP unavailable	Unavailable

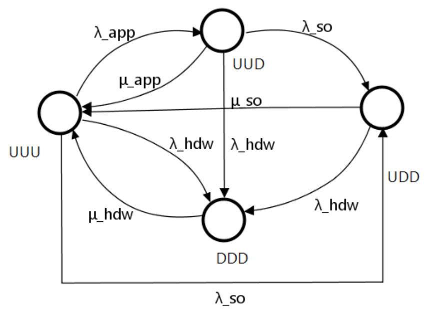


Figure 5: CTMC – Moodle.

In this CTMC model, λ represents the failure rate and μ the repair rate of the system components. Their values, incorporating the acceleration factor discussed in Section 4, are presented in Table 7. The states follow a sequence where U represents UP and D represents DOWN, corresponding to Hardware, OS, and APP.

The system's availability is the probability of being in the UUU state, represented as $A_s = \pi_{(UUU)}$, where $\pi_{(UUU)}$ is the steady-state probability of being in the UUU state. In this state, the system is available. If a failure occurs in any LAMP component with a rate λ_{app}, the system transitions to the UUD state. From this state, the system can be repaired at a rate μ_{app}, returning to the UUU state. A failure in the operating system with rate λ_{os} sends the system to the UDD state, which can be repaired at a rate μ_{os}. Hardware failures with rate λ_{hdw} can occur from either UUD or UDD, leading to the DDD state, which can only be restored by a hardware repair at a rate μ_{hdw}.

The availability model of the system can be evaluated using Equation 1, which was generated using the Mathematica tool [8] in conjunction with the Mercury tool [16].

$$A = \frac{(\lambda_{\text{hdw}} + \lambda_{\text{os}} + \mu_{\text{app}})\, \mu_{\text{hdw}}\, (\lambda_{\text{hdw}} + \mu_{\text{os}})}{(\lambda_{\text{app}} + \lambda_{\text{hdw}} + \lambda_{\text{os}} + \mu_{\text{app}})\, (\lambda_{\text{hdw}} + \mu_{\text{hdw}})\, (\lambda_{\text{hdw}} + \lambda_{\text{os}} + \mu_{\text{os}})} \tag{1}$$

The steady-state probability can be computed by solving the linear system composed of balance equations, as well as by using the sum of probability properties.

4 Case Study

In this section, we present a study designed to validate and evaluate the architecture proposed in Section 2, aiming to verify the suitability of the availability models discussed in Section 3 about real-world scenarios. We present three case studies that demonstrate the practical application of the proposed model in detail. Initially, we explore the validation process and the adopted experimental approach, presenting the methods used to guarantee the accuracy of the model and the initial results obtained from the simulations. Next, we analyze the system's availability, focusing on its ability to remain operational. Finally, we conduct a sensitivity analysis to quantify the impact of different configurations on model performance. We offer a sensitivity ranking and graphs of parametric variations, providing detailed insights into each parameter's influence and facilitating model optimization.

4.1 Case study I – Validation and experimental approach

To validate the proposed availability architecture, our analysis involved a detailed study of the server log file, which tracks the status of the Moodle server. This file contains a sequence of entries with timestamps and server status every five seconds, providing essential data for the analysis. By examining all instances of Moodle's availability and unavailability, we were able to calculate the Mean Time to Failure (MTTF) – the average time between two consecutive failures, and the Mean Time to Repair (MTTR) – the average time required for the system to recover from a failure.

Due to the high failure time values, a case study for validation becomes infeasible without adjustments. The works in [4], [5], and [6] highlight this challenge in validation scenarios and demonstrate the need for using an "acceleration factor" for components with high failure times. Our study applied an acceleration factor of 1000 time units to the MTTF, resulting in the values shown in Table 7.

Table 2: Input Parameters – Acceleration Factor

Component	MTTF (h)	MTTR (h)
Hardware	8.76	1.67
Operational System	2.88	1
MySQL	1.44	1
Apache	0.7884	0.5
PHP	0.7884	0.5

The Moodle server was monitored for six days, yielding substantial data for analysis. During this period, the log file continuously recorded the server's status. The server experienced 94 failures and recoveries. To calculate the real MTTF, the acceleration factor was undone by reducing the component parameters by the same factor of 1000. The resulting Mean Time to Failure (MTTF) was 332,672 hours, and the Mean Time to Repair (MTTR) was 1.152 hours. We calculated an availability (A) of $A = 0.996583$.

Adopting the validation method proposed by [10], we calculated the confidence interval for availability based on the number of failures and repairs, incorporating a degree of freedom to derive an F distribution closely matching the collected data. Applying the test scenario values to the modeled environment, we arrived at the results in Table 3.

Table 3: Base values for degree of freedom

F Distribution	Value
Degree of freedom	94
Lower critical value – L	0.6658
Upper critical value – U	1.502

From the lower and upper critical values, we calculated the availability confidence interval [10] as follows:

$$\left(\frac{1}{1+U}, \frac{1}{1+L}\right) = \left(\frac{1}{1+1.502}, \frac{1}{1+0.6658}\right) \tag{2}$$

The calculated lower and upper confidence intervals (CI) were $(0.99487, 0.99772)$. We calculated the average failure, repair, and availability times for the experiment. We obtained an availability of 0.99658. Comparing these results with the models proposed in Figure 5, we found an availability of 0.99750, as shown in Table 4.

Table 4: Experiment Confidence Interval

CI - 95%	Experiment	Model
(0.99487 – 0.99772)	0.99658	0.99750

The data from the Moodle server falls within the 95% confidence interval of the RBD and CTMC models for availability and related metrics. This alignment validates the test environment, confirming that it can be used to develop new study models and obtain other metrics.

4.2 Case study II – Availability Evaluation

To evaluate the case study, we calculated the system availability using the MTTF and MTTR values of the components, expressed in hours, as shown in Table 5, based on studies by [4], [5], and [6].

The metrics for this evaluation were obtained using the Mercury tool. The calculated metrics include MTTF, MTTR, availability, number of nines, uptime, and downtime. Table 6 summarizes the results for the initial model.

Table 5: System Components

Component	MTTF (h)	MTTR (h)
Hardware	8760	1.67
Operational System	2880	1
MySQL	1440	1
Apache	788.4	0.5
PHP	788.4	0.5

Table 6: RBD Model Results

Metric	Value
MTTF (hours)	270.81150
MTTR (hours)	0.67786
Availability (%)	0.99750
Number of 9's	2.60260
Uptime (hours in the year)	8743.92581
Downtime (hours in the year)	21.88695

Mercury was also used to evaluate the CTMC model. To calculate the metrics, we determined each component's failure and repair rates. Table 7 presents these values, calculated as the inverse of failure and repair times.

Table 7: CTMC Input Parameters

Rate	Description	Value h^{-1}
λ_{hdw}	Hardware Failure Rate	1/8760
λ_{os}	Software Failure Rate	1/2880
λ_{app}	Application Failure Rate	1/0.30948
μ_{hdw}	Hardware Repair Rate	1/1.67
μ_{os}	Software Repair Rate	1
μ_{app}	Application Repair Rate	1/0.607

The results obtained for the CTMC model are shown in Table 8.

Table 8: CTMC Model Results

Metric	Value
Availability	0.99750
Downtime (hours per year)	21.86728
Unavailability	0.00249
Uptime (hours per year)	8738.13271
Number of 9's	2.60270

4.3 Case study III – Sensitivity analysis

In this case study, we conducted a sensitivity analysis [14] using the CTMC model (Figure 5). The differential sensitivity analysis, characterized by the sensitivity index $S_\lambda(Y)$, indicates the impact of parameters λ and μ on system availability. Table 9 shows this analysis's results, ranking each parameter's sensitivity.

Table 9: Sensitivity Analysis Results

Parameter	Sensitivity Value
λ_{app}	-1.9597×10^{-3}
μ_{app}	1.9592×10^{-3}
μ_{os}	3.4702×10^{-4}
λ_{os}	-3.4665×10^{-4}
μ_{hdw}	1.9060×10^{-4}
λ_{hdw}	-1.9043×10^{-4}

Sensitivity analysis identified the application (APP) as the most critical component for system availability, particularly in terms of failure and repair rates (λ_{app} and μ_{app}). Optimizing these parameters can significantly improve system reliability. Figures 6 and 7 show the variation of availability about $MTTF_{app}$ and $MTTR_{app}$, respectively.

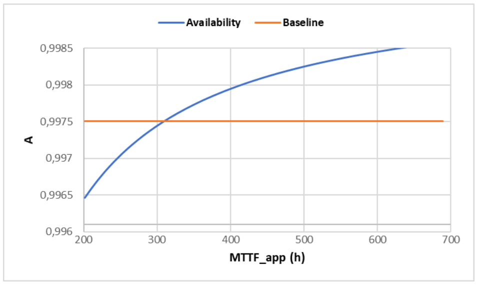


Figure 6: Variation of A vs $MTTF_{app}$

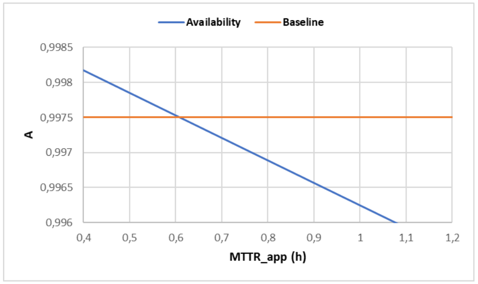


Figure 7: Variation of A vs $MTTR_{app}$

Similar behavior is observed in the operating system repair rate (μ_{os}) shown in Figure 8, where increasing the repair time decreases system availability.

Finally, we analyzed the influence of the operating system failure rate (λ_{os}) in Figure 9, noting diminishing returns in availability improvement as $MTTF_{os}$ increases.

Sensitivity analysis is fundamental in identifying critical components that impact system availability, enabling resource optimization and strategic decision-making for maintenance and system improvement.

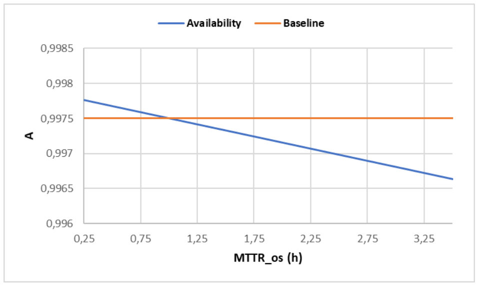


Figure 8: Variation of A vs $MTTR_{os}$

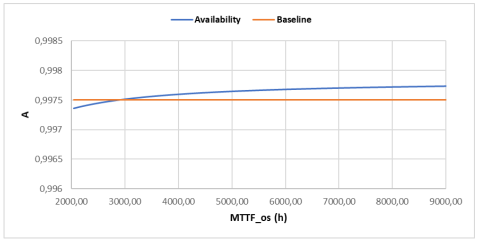


Figure 9: Variation of A vs $MTTF_{os}$

5 Conclusion

It was imperative to explore the implementation of Moodle, one of the leading LMS platforms, to create and study an availability model. This was a crucial step in our research as it allowed us to define the main parameters of interest through techniques that measure its ability to provide the service correctly.

Based on previous studies that separately measured failure and repair times for LAMP stack components, hardware, and software, it was initially necessary to create two availability models using RBD and CTMC.

We also performed a sensitivity analysis to identify the metrics that most impact system availability. We focused on failure rates (λ_{os} and λ_{app}) and repair rates (μ_{os} and μ_{app}), as the parametric sensitivity analysis indicated that these rates are the most influential in the system. We deepened the investigation through graphs that examined the relationship between system availability and these failure and repair rates. This analysis highlighted the system components that require special attention to improve availability.

Finally, validating the proposed models and specifically adopting a model based on CTMC aids in proposing scalable future architectures, as it can represent part of the components or the entire system. In future work, metrics such as the associated cost can also be calculated.

6 Acknowledgment

This work was supported by Project No. 7059 - Neuraspace - AI fights Space Debris, reference C644877546-00000020, supported by the RRP - Recovery and Resilience Plan and the European Next Generation EU Funds, following Notice No. 02/C05-i01/2022, Component 5 - Capitalization and Business Innovation - Mobilizing Agendas for Business Innovation, and within the scope of CISUC R&D Unit - UIDB/00326/2020.

References

[1] Ajlan Al-Ajlan and Hussein Zedan. 2008. Why moodle. In *2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems*. IEEE, 58–64.

[2] Hakan Altınpulluk and Mehmet Kesım. 2021. A systematic review of the tendencies in the use of learning management systems. *Turkish Online Journal of Distance Education* 22, 3 (2021), 40–54.

[3] Jean Araujo, Vandi Alves, Danilo Oliveira, Pedro Dias, Bruno Silva, and Paulo Maciel. 2013. An Investigative Approach to Software Aging in Android Applications. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*. 1229–1234. https://doi.org/10.1109/SMC.2013.213

[4] Jamilson Dantas, Rubens Matos, Jean Araujo, and Paulo Maciel. 2012. An Availability Model for Eucalyptus Platform: an analysis of warm-standy replication mechanism. *ACM. IEEE Computer Society* 3, 3 (2012), 1664–1669.

[5] Igor de Oliveira Costa. 2015. *Modelos par análise de disponibilidade em uma plataforma de mobile backend as a service*. Ph. D. Dissertation. Universidade Federal de Pernambuco.

[6] Maria Clara dos Santos Bezerra. 2015. *Modelos para análise de disponibilidade de arquitetura de um serviço de Vod Streaming na nuvem*. Ph. D. Dissertation. Universidade Federal de Pernambuco.

[7] Sithara HPW Gamage, Jennifer R Ayres, and Monica B Behrend. 2022. A systematic review on trends in using Moodle for teaching and learning. *International journal of STEM education* 9, 1 (2022), 9.

[8] Wolfram Research, Inc. [n. d.]. Mathematica, Version 14.1. https://www.wolfram.com/mathematica Champaign, IL, 2024.

[9] V. D. Jogi and A. Sinha. 2016. Performance Evaluation of MySQL, Cassandra and HBase for Heavy Write Operation. In *3rd Int'l Conf. on Recent Advances in Information Technology (RAIT-2016)*. Dhanbad, India. https://doi.org/10.1109/RAIT.2016.7507948

[10] W. R. Keesee. 1965. A Method of Determining a Confidence Interval for Availability. Point Mugu, California: Miscellaneous Publication.

[11] Alena Thomas Joseph Kurien, Sharon Ann Mathew, and Suja Cherukullapurath Mana. 2022. Development of PHP and MySQL based Digital Asset Management System for Secure Organizations. In *2022 6th International Conference on Trends in Electronics and Informatics, ICOEI 2022 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 1859–1863. https://doi.org/10.1109/ICOEI53556.2022.9776698

[12] Paulo Romero Martins Maciel. 2023. *Performance, Reliability, and Availability Evaluation of Computational Systems, Volume I: Performance and Background*. Chapman and Hall/CRC.

[13] Ronierison Maciel, Jean Araujo, Jamilson Dantas, Carlos Melo, Erico Guedes, and Paulo Maciel. 2018. Impact of a DDoS attack on computer systems: An approach based on an attack tree model. In *2018 Annual IEEE International Systems Conference (SysCon)*. 1–8. https://doi.org/10.1109/SYSCON.2018.8369611

[14] Ronierison Maciel, Jean Araujo, Jamilson Dantas, Carlos Melo, Erico Guedes, and Paulo Maciel. 2020. Bottleneck Detection in Cloud Computing Performance and Dependability: Sensitivity Rankings for Hierarchical Models. *Journal of Network and Systems Management* 28 (2020), 1839–1871. https://doi.org/10.1007/s10922-020-09562-9

[15] Moodle. [n. d.]. Moodle [online]. https://docs.moodle.org/403/en/About_Moodle. Accessed: April 02, 2024.

[16] Bruno Silva, Rubens Matos, Gustavo Callou, Jair Figueiredo, Danilo Oliveira, Joao Ferreira, Jamilson Dantas, Aleciano Lobo, Vandi Alves, and Paulo Maciel. 2015. Mercury: An Integrated Environment for Performance and Dependability Evaluation of General Systems. In *INDUSTRIAL TRACK AT 45TH DEPENDABLE SYSTEMS AND NETWORK CONFERENCE (DSN 2015)*.

[17] P. Wannapiroon, N. Kaewrattanapat, and J. Premsmith. 2019. Development of Cloud Learning Management Systems for Higher Education Institutions. In *2019 Research, Invention, and Innovation Congress (RI2C 2019)*. Bangkok, Thailand. https://doi.org/10.1109/RI2C.2019.8959675