
**UNIVERSITY OF PETROȘANI**
**DOCTORAL SCHOOL**

# DOCTORAL THESIS

DOCTORAL ADVISOR:
Prof. univ. dr. Marius Cioca

DOCTORAL STUDENT:
Schuszter Ioan Cristian

**2024**

**UNIVERSITY OF PETROȘANI**
**DOCTORAL SCHOOL**

# USING MACHINE LEARNING AND BIG DATA METHODS TO IMPROVE THE QUALITY OF CRITICAL SYSTEMS

DOCTORAL ADVISOR:
Prof. univ. dr. Marius Cioca

DOCTORAL STUDENT:
Schuszter Ioan Cristian

**2024**

# 1 Abstract

This doctoral thesis explores critical system architectures, aiming to improve their quality and reliability through exhaustive research and innovative methodologies. Spanning 119 pages and structured into six chapters, this work embarks on a journey of exploration, analysis, and implementation, with the final goal of advancing critical system engineering.

**Chapter 1** provides a comparative study of existing system architectures, offering insights into their strengths and weaknesses. This lays the groundwork for further explorations, providing fundamental knowledge.

**Chapter 2** is the literature overview section of this thesis and it looks into various facets of systems engineering, machine learning techniques, and sustainability in software systems. It begins with a comprehensive review covering reliable systems engineering, text classification methodologies in machine learning, classical machine learning techniques, and modern (deep) machine learning methods. Additionally, it explores sustainable software systems, highlighting the significance of environmentally conscious software practices.

Further, the comparative studies section scrutinizes the intersection of reliability and sustainability in systems engineering, investigates intelligent real-time monitoring approaches for computing systems, and evaluates the application of machine learning in fault prediction and sustainability efforts.

**Chapter 3** explores new methodologies and technologies to improve the quality of critical systems, relying on a multidisciplinary approach. It investigates innovative research methods and architectures to address evolving challenges.

**Chapter 4** shifts the focus towards reliability techniques applied to software engineering methodologies. Various strategies for enhancing reliability in critical system development are examined from the perspective of the software engineers involved in creating these systems.

**Chapter 5** investigates efficient monitoring techniques for managing critical software systems. By integrating advanced monitoring tools and methodologies, it aims to provide engineers with insights for smooth system operation.

**Chapter 6** focuses on the development, implementation, and validation of an ensemble for improving the quality of critical systems using machine learning techniques and big data. This chapter explores data-driven insights and predictive analytics to offer innovative solutions.

Finally, **Chapter 7** concludes the thesis by synthesizing key findings, highlighting personal contributions, and offering insights into future research directions. It serves as a testament to the author's dedication to advancing critical system engineering.

Human dependence on technology has become increasingly pronounced in recent years, reaching a point where much of our lives rely on interaction with software, with a recent example being the dependence of education and businesses on video conferencing software like Zoom during the global COVID-19 epidemic in 2020. A disruption to any of these critical services can have a very negative impact on the economy in a short amount of time. For these reasons, finding and implementing solutions based on big data and machine learning technologies to improve the

reliability of ”critical” systems is an important target, which this thesis aims to achieve.

# 2 Introduction

Today, software systems are often composed of various components written in a wide range of programming languages, depending on the purpose of the software, and the challenge of monitoring their quality is exacerbated due to their distributed nature and the simplicity with which this is achieved using a ”microservices” design pattern and the proliferation of process isolation (containerization) or orchestration technologies. Another primary need, often implemented inefficiently, is the uniform monitoring of all these components, identifying defects as quickly as possible, and subsequently developing real-time predictive capabilities to avoid defects from the outset. There are already some works addressing the application of artificial intelligence methods, particularly natural language processing, to identify anomalies in the logs of certain systems, but essentially performed on synthetic systems.

This thesis aims, initially, to identify the landscape and modern components necessary for building a reliable software system, with a case study conducted on a real Single Sign-On and identity management system at CERN used in production. Additionally, the organization is part of a federation of research institutes and universities from all over the world, making the system a critical link in this entire mechanism. The system can be considered critical because it has approximately 16,000 daily users who depend on it to carry out their activities. Open-source technologies are specifically targeted, which can be used to build a system without fear of various aspects such as intellectual property that need to be considered or the costs paid for licenses.

The thesis is divided into 3 different parts, each focusing on a facet of analyzing reliable systems and improving them using modern big data and machine learning techniques. The 3 parts are developed in the following paragraphs:

## 2.1 Study on Machine Learning and Big Data Methods

This part of the thesis aims to explore existing literature on reliable systems, preparing the reader in the field by exposing the main ideas existing in the industry, as well as the mechanisms through which the reliability of systems can be influenced. Distributed systems and the modern methods by which they can be composed are discussed. Next, the importance of related monitoring and alerting systems is presented, which are used to have an overview of distributed systems that need to be put in place to ensure that we achieve a satisfactory level of reliability for a system running in production without major interruptions in its operation.

Finally, we move on to studying various machine learning methods through which data produced by a system and the monitoring system (often textual) can be used to predict future issues that may be encountered in such a system. The ultimate goal of the thesis is this: to describe the ideal software architecture for a system considered critical (such as those for electricity or water supply), and the last lever we can use is these predictive models that allow us to anticipate

and correct the system before it fails. The best implementation methods for these systems are identified through the use of case studies, where various compromises in the development of these systems are highlighted in the light of research.

## 2.2 Exploratory Research, Methods, and Techniques for Improving the Quality of Critical Systems

The second part of the thesis relies more on exploratory research into what makes a critical distributed system superior to others, which often suffer from scalability issues, are cumbersome, and self-heal too slowly or not at all. Specifically, we can divide this section into 3 more granular ones. First, we focus on how to develop the lowest level in a reliable system: the data layer. Various methods for achieving reliability at the database level are presented, especially in the case of relational databases, which are notoriously difficult to scale beyond a certain level. A case study is conducted, and major shortcomings are identified when the storage system is distributed across multiple databases. The next research direction consists of non-relational databases, especially those that store data in memory for fast access. An implementation of a non-relational database based on a programming model called the ”actor model” is developed and compared with Redis, one of the most used implementations of this type of database in the IT community. The performance of the developed system is significantly superior, as can be seen in the resulting graphs.

Another important point of discussion is how the software services developed by the team can be deployed into production as quickly as possible without being disruptive to the daily operation of the services and without users experiencing service degradation (considering that we are dealing with a system that focuses on logging in 16,000 users every day). Openshift is presented as a solution for deploying services, with a load balancer placed in front of a service so that for a client, it has a single entry point, when in fact behind the load balancer, there is a potentially unlimited number of identical servers.

Once the issue of storage systems and technologies that allow us to scale the system infinitely is addressed, we turn our attention to massive monitoring systems, which process a huge amount of data in near real-time, so that any defects can surface as quickly as possible and not affect the organization's SLAs (Service Level Agreements). A series of technologies and implementation methods are presented, aiming to choose the ideal architecture and system that can process a sufficient amount of data without issues. The choice of a log ingestion architecture based on the open-source technology Flume is justified, then the use of Prometheus and Grafana to create alerts and dashboards to have an overview of the entire range of services. The developed system is deployed for the production system at CERN and provides alerts and a continuous source of data regarding the operation of various software services in the system. A case study is presented, focusing on the speed of the system's recovery from an abnormal situation.

## 2.3 Development, Implementation, and Validation of a Quality Improvement System for Critical Systems Based on Machine Learning Models and Big Data

The final section of this work focuses on predictive systems, more precisely the methods and techniques that can be used so that the performance of a system can be predicted before it suffers from abnormal behavior. The problem starts from data analysis, their collection, and storage in a format that allows their use for training a prediction algorithm. The proposed system is based on data ingestion from the Prometheus system, so that the change in processor usage for a certain service is aggregated at intervals of 5 minutes. The analysis system must process this information and propose an approximate value at which the service will be used in the next 5 minutes, based on all historical data. For the complete development and evaluation of the system, a dataset collected over a month at CERN is used.

Three architectures based on machine learning and deep neural networks are proposed for implementing the system: CNN, RNN, and LSTM. At the same time, to ensure that the results are conclusive, these implementations are compared with a system based on classical methods of analyzing time-series data collections, as they are known: ARIMA. After evaluating the neural networks developed for this system, RNN emerges as an obvious winner and is compared with various heuristics that could be used for the same purpose, but it is again an obvious winner.

A final detailed analysis of the work is based on the possibility of using this system in an innovative way to predict the CPU usage of services and the benefits in terms of system sustainability using this machine learning method to provide more or fewer servers as needed. In realistic terms, an energy saving of approximately 60% can be observed when these algorithms are used, which is a very important result.

# 3 Personal Contributions

The following significant personal contributions are highlighted to emphasize the innovative concepts proposed in this thesis:

1. Proposing architectural considerations for a critical production service used at CERN, one of the largest research institutions in the world. The work elucidates the overall architectural framework and key components of such a scalable system, recommending specific technologies for its implementation. Additionally, it addresses the trade-offs involved in transitioning from a monolithic architecture to a distributed fault-tolerant one. (Chapter 2,3)

2. Evaluating and selecting a suitable monitoring system component, such as a REST API, for deployment alongside each monitored system element. This component was tailored to ingest and interpret log data in diverse formats, ensuring efficient transmission for

archiving and processing while emphasizing scalability to manage potential surges in log messages, as it is a big data domain. (Chapter 3)

3. Conducting a comparative assessment of Apache Flume and Logstash as candidate software solutions for log ingestion and processing. Following the analysis, Apache Flume emerged as the preferred choice due to its reduced memory usage, more flexible licensing terms, and performance comparable to that of Logstash. (Chapter 3)

4. Identifying an unaddressed issue concerning the management of complexity post-deployment, particularly in meeting required Service Level Agreements (SLAs). The anticipated challenge revolves around the increased time and effort required for accessing service logs to diagnose error root causes. While not explored in this paper, this problem is commonly encountered in distributed architectures. Future research endeavors will explore potential solutions. (as identified in **Chapter 3**)

5. Implementing an efficient NoSQL database using the actor model. This database design approach leverages its concurrency and scalability features to enhance performance and resilience, offering a promising solution for managing large-scale data in distributed systems. (as proposed in **Chapter 3**)

6. Implementation of an automatic resource usage prediction system, facilitated by establishing a robust monitoring infrastructure. This system enables the collection of high-quality data essential for scaling operations effectively. (as identified in Chapter 5)

7. Development of various machine learning and deep learning algorithms aimed at validating the hypothesis of efficient resource usage prediction. (Chapter 6)

8. Integration of advanced anomaly detection mechanisms, enhancing the system's ability to identify and mitigate potential issues before they escalate.

9. Deployment of real-time data analytics frameworks, enabling rapid decision-making and optimization of resource allocation strategies.

Additionally, in Chapter 4, more "human" aspects of the development process have been considered, as human resources are also part of the system. Therefore, some key findings and contributions we proposed were:

1. Illustrated the considerable influence of robust metrics and established methodologies such as SAFe on team performance and predictability within critical software systems.

2. Emphasized the importance of harmonizing development endeavors with organizational objectives by reaching consensus on key business metrics among team members.

3. Highlighted the benefits of using automated tools to streamline support efforts and pave the way for seamless integration of future changes, reducing concerns regarding regression.

4. Developed automated tools providing instantaneous feedback for coding activities, thereby enhancing efficiency and reliability in software development workflows.

5. Advocated for efficient team management practices alongside the incorporation of code reviews and automated tests to bolster predictability and minimize team workload.

6. Advanced the comprehension and implementation of methodologies and technologies aimed at fortifying the reliability of critical software systems within systems engineering.

# 4 Research Papers Published During the Doctoral Studies

During the research for this doctoral thesis, I contributed to the development of 11 distinct papers as an author or co-author, of which I was the main author for 7 of them.

It is worth mentioning that 4 of these papers were published in ISI-ranked journals, and in 3 of them, I served as the primary author. The remaining papers were published in conference proceedings indexed in the Web of Science or other databases.