Kampa, S. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 7, July-2023, pg. 01-06.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

System and Software Requirements in Relation to Observability and Explainability

Sandeep Kampa*
Member, IEEE, United States
E-mail: sandeepkampa5@gmail.com

Received date: 03 June 2023**,** Manuscript No. ijcsma-23-101176; **Editor assigned:** 04 June 2023**,** Pre QC No ijcsma-23-101176 (PQ); **Reviewed:** 19 June 2023, QC No. ijcsma-23-101176 (Q)**; Revised:** 25 June 2023**,** Manuscript No. ijcsma-23-101176 (R); **Published date:** 30 June 2023
DOI. 10.5281/zenodo.8371766

Abstract

Software maintenance and evolution are crucial aspects of software development. In today's world, observability and explainability are becoming essential requirements for software systems. This research paper investigates the relationship between software maintenance and evolution with observability and explainability. The paper explores the importance of observability and explainability in software systems and how they impact the maintenance and evolution of software systems. The research paper presents various techniques and tools for achieving observability and explainability in software systems. The paper also highlights the challenges and future research directions in the field of software maintenance and evolution in relation to observability and explainability.

Keywords*:* Observability and detectability; Infrastructure systems; Cloud computing; Cloud monitoring; Software development; Practice computer applications

1. Introduction

Software maintenance and evolution are crucial aspects of software development. As software systems evolve and

ISSN: 2321-8363
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

grow, they require constant updates and maintenance to meet the changing demands of the users. The maintenance and evolution of software systems involve the modification of existing software systems to meet new requirements, fix bugs and errors, and improve performance and reliability. However, as software systems become more complex and distributed, it becomes increasingly challenging to maintain and evolve them.

Observability and explainability are two critical requirements for modern software systems. Observability refers to the ability to observe and monitor the behavior and performance of software systems in real-time. Observability allows developers to identify and fix issues before they become critical problems. Explainability, on the other hand, refers to the ability to explain how the software system behaves and why it behaves in a particular way. Explainability helps developers understand the behavior of the software system, which is essential for maintaining and evolving the software system.

In this research paper, we investigate the relationship between software maintenance and evolution with observability and explainability. We explore the importance of observability and explainability in software systems and how they impact the maintenance and evolution of software systems. We present various techniques and tools for achieving observability and explainability in software systems. We also highlight the challenges and future research directions in the field of software maintenance and evolution in relation to observability and explainability.

2. Importance of Observability in Software Systems Maintenance and Evolution

Observability allows developers to monitor the behavior and performance of software systems in real-time. Observability helps developers identify and fix issues before they become critical problems. Observability also helps developers understand the behavior of the software system, which is essential for maintaining and evolving the software system.

Observability refers to the ability of software engineers to observe and monitor the behavior of software systems in real-time. Observability is critical in software maintenance and evolution because it helps software engineers to identify and fix problems quickly. Observability involves using monitoring tools and techniques to collect data on the behavior of software systems, which can then be analyzed to identify issues and optimize performance. Observability refers to the ability of software engineers to observe and monitor the behavior of software systems in real-time. Observability is critical in software maintenance and evolution because it helps software engineers to identify and fix problems quickly. Observability involves using monitoring tools and techniques to collect data on the behavior of software systems, which can then be analyzed to identify issues and optimize performance [1].

One of the primary tools used in observability is log data. Log data is a record of all the activities that occur in a software system, including errors, exceptions, and other events. Software engineers can use log data to identify the cause of errors and other issues in a software system. The use of log data in observability has been studied

ISSN: 2321-8363
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

extensively, and several approaches have been proposed to improve the effectiveness of log data analysis [2]. Another tool used in observability is tracing. Tracing involves tracking the flow of requests and responses through a software system to identify performance bottlenecks and other issues. Tracing has been shown to be an effective technique for identifying the root cause of performance problems in distributed systems [3].

3. Importance of Explainability in Software Systems Maintenance and Evolution

Explainability, on the other hand, is critical for understanding the behavior of the software system. Explainability helps developers understand why the software system behaves in a particular way. Understanding the behavior of the software system is essential for maintaining and evolving the software system. Explainability also helps developers identify the root cause of issues and fix them effectively.

Explainability refers to the ability of software engineers to understand why a software system behaves in a particular way. Explainability is essential in software maintenance and evolution because it helps software engineers to make informed decisions about changes to a software system. Explainability involves using techniques such as model interpretation, visualization, and natural language explanations to help software engineers understand the behavior of software systems [4].

One of the primary techniques used in explainability is model interpretation. Model interpretation involves analyzing the internal workings of a machine learning model to understand how it makes predictions. Model interpretation has been shown to be an effective technique for improving the explainability of machine learning models [5].

Another technique used in explainability is visualization. Visualization involves representing data in a visual format, which can help software engineers to identify patterns and relationships in the data. Visualization has been shown to be an effective technique for improving the explainability of complex software systems [4].

4. Techniques and Tools for Achieving Observability and Explainability in Software Systems

The increasing complexity of modern software systems has made it challenging for engineers to monitor and diagnose issues that may arise in real-time. Observability is the ability to understand and analyze the internal state of a system based on external inputs. It involves collecting, aggregating, and analyzing data from various sources, such as logs, metrics, and traces. This data can provide insights into the behavior and performance of the system and help engineers diagnose issues quickly. In recent years, the need for observability has led to the development of a variety of tools and techniques. These tools can help engineers monitor the performance and behavior of their software systems in real-time, and they can also provide insights into the root cause of issues. In this paper, we provide a comprehensive review of the different tools that can be used to achieve observability of software systems [6].

Kampa, S. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 7, July-2023, pg. 01-06.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

There are various techniques and tools for achieving observability and explainability in software systems. Some of the commonly used techniques and tools are,

4.1 Logging

Logging is a technique used for capturing and storing the behavior and performance of software systems. Logging allows developers to observe and monitor the behavior of software systems in real-time. Logging also provides a historical record of the behavior of the software system, which is useful for analyzing the behavior of the software system.

4.2 Tracing

Tracing is a technique used for capturing the flow of requests and responses in software systems. Tracing allows developers to understand the behavior of software systems and identify the root cause of issues.

4.3 Metrics

Metrics are measurements of the behavior and performance of software systems. Metrics allow developers to monitor the behavior and performance of software systems in real-time. Metrics also provide a historical record of the behavior and performance of software systems, which is useful for analyzing the behavior and performance of the software system.

4.4 Profiling

Profiling is the process of measuring the performance of individual components of a system. Profiling can be used to identify bottlenecks and optimize performance

5. Benefits of Observability Tools

Observability tools provide several benefits, such as,

5.1 Real-time Monitoring

Observability tools provide real-time monitoring of software systems, enabling engineers to quickly identify and diagnose issues.

5.2 Performance Optimization

Observability tools can help engineers optimize the performance of software systems by identifying bottlenecks and areas for improvement.

5.3 Root Cause Analysis

Kampa, S. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 7, July-2023, pg. 01-06.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

Observability tools can provide insights into the root cause of issues, enabling engineers to fix issues quickly and efficiently.

6. Limitations and Challenges

Observability tools also have limitations and challenges, such as,

6.1 Data Overload

Observability tools can generate large amounts of data, making it challenging to manage and analyze the data effectively.

6.2 Complexity

Observability tools can be complex to set up and configure, requiring specialized knowledge and skills.

6.3 Cost

Some observability tools can be expensive to purchase and maintain, making it challenging for smaller organizations to implement.

7. Conclusions

Observability and explainability are essential requirements for modern software systems. Relationship between Observability, Explainability, and Software Maintenance and Evolution: Observability and explainability are closely related to software maintenance and evolution. Observability helps software engineers to identify and fix problems quickly, while explainability helps software engineers to make informed decisions about changes to a software system. By combining observability and explainability, software engineers can gain a deep understanding of the behavior of a software system and make informed decisions about its maintenance and evolution. Observability and explainability are critical concepts in software maintenance and evolution. By using monitoring tools and techniques to collect data on the behavior of software systems and using techniques such as model interpretation and visualization to understand that data, software engineers can gain a deep understanding of the behavior of software systems and make informed decisions about their maintenance and evolution.

References

[1] Heer, J., Viégas, F. B., Wattenberg, M "Voyagers and voyeurs: supporting asynchronous collaborative information visualization." *Proc. SIGCHI conf. Hum. factors comput. Syst* (2007).

[2] Liu, Y., et al. "A survey of data center network architectures, topologies." *Donald Bren Sch. Inf. Comput. Sci* (2017).

[3] Arrieta, A.B, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

challenges toward responsible AI." *Inf. fusion* 58 (2020): 82-115.

[4] Chuang, J., Manning, C. D., Heer, J. "Termite: Visualization techniques for assessing textual topic models." *Proc. int. work. conf. adv. vis. interfaces* (2012).

[5] Ribeiro, M. T., Singh, S., Guestrin, C. "Model-agnostic interpretability of machine learning." *arXiv prepr* (2016).

[6] Liu, S., et al. "Towards better analysis of machine learning models: A visual analytics perspective." *Vis. Inform* 1.1 (2017): 48-56.