Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2021, 8(1):264-269**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

A Guide to Troubleshooting in Unfamiliar Environments

Mohit Thodupunuri

MS in Computer Science
Sr Software Developer - Charter Communications Inc
Email id: Mohit.thodupunuri@gmail.com

Abstract: With the evolution of technology, there comes the need for optimized solutions but with the use of limited resources as the competition is increasing in every field of life and more promising in information technology. The unfamiliar environments may need to be configured by the developers where they face serious challenges. These new environments are required to be quickly adapted to compete in the market for both individuals and organizations. Thus, to keep the performance and to avoid disruptions in the system. However, there exist certain challenges with the adoption of these unfamiliar methods. They can range from a lack of background knowledge to complex implementations. These challenges are addressed in the writing and some strategies are proposed as a guide. These practices are required to be followed for the effective use of newly announced systems or any unfamiliar environment that is not common in the industry.

Keywords: troubleshooting, documentation, environments, modern systems, useful methods, challenges, strategies

1. Introduction

Technology nowadays is more competitive where providing the solutions to a problem is not sufficient but requires extra care. These simple approaches were good for the past but now evolution has reached a point where optimization is associated with the solutions. Resources need to be saved, and more efficient methods are therefore required to be implemented in a short time. This is a challenge for the developers and the organizations as troubleshooting new or unfamiliar systems is complex. The quick adoption of these environments is necessary, and the performance of the system should not be compromised.

The given writing therefore addresses the significance of utilizing modern environments and the challenges that appear due to the use of unfamiliar systems. The challenges may include restricted resources by the provider, limited background knowledge, handling system failures, and lack of communication between the communities of the system. These challenges however are addressed considering their importance for developers or organizations. These appropriate strategies help to better utilize the latest methods and avoid any conflicts that can arise during the integration with large applications.

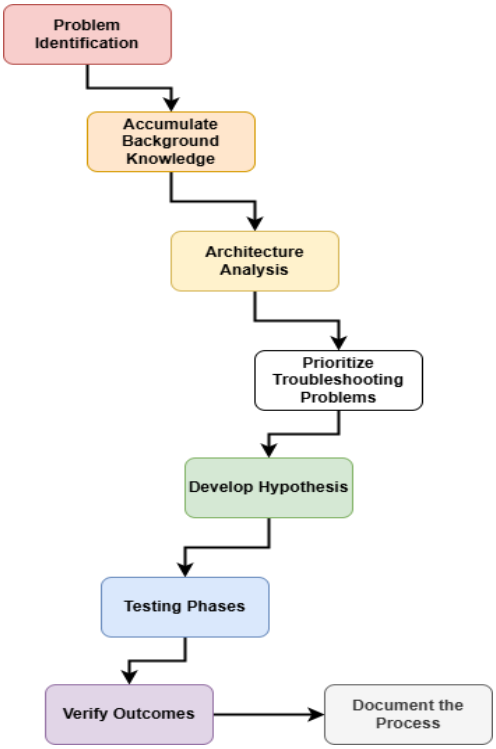

Figure 1: Troubleshooting Process for Unfamiliar Environments

The above figure 1 shows the complete flow of mandatory steps that are recommended while working in unfamiliar environments. It's important to gather all the relevant knowledge as a first step in working on a problem. The architecture of the system once studied makes it easy to move on to the development phase. However, the critical steps need to be prioritized before starting the implementation process. The solution once developed can be tested and validated against the required results. The results can then be verified and at the end added to the documentation to maintain the record for future reference [1].

2. Literature Review

The Root Cause Analysis (RCA) helps deploy a modification or addition in the system in a component-based deployment. Here, the environment is processed in an iterative way where each component is responsible for its working. In this way, the system can handle the resources and the functionalities separately and that is easy to manage. The root cause is analysis methods like RCSF help to monitor applications by anomaly detectors. The disturbed sequences or chunks are identified beforehand using these kinds of methods [2].

Adaptability is important on both individual and organizational levels. The appropriate strategies are required for an individual to adapt new development methods to better survive in the industry. Similarly, organizations have to hire more people who are adaptive to advanced or unfamiliar methods based on their previous expertise [3]. However, adaptability is equally important for the software. Each of the components of the software can be monitored by following the flow between different clusters [4].

The iterative integration makes it easy for the developers to make changes in the code in a controlled manner. In this way, the system is not largely affected immediately and any potential problems that arise during the process or after implementation can be easily identified and resolved [5]. As the testing of the system is equally important to provide the best possible product to the users, the iteration starts with the initial integrations and ends with the unit testing. Following the process, the cycle is repeated, and the system testing is included for further identifications and resolutions [6].

3. Problem Statement

The software industry is continuously evolving, and the competition now lies in the utilization of modern techniques to survive in a competent environment where fewer resources are required to achieve the maximum

results in a short interval of time. The best solution to save the organizational efforts is the use of the latest environments that are suitable for the system. However, it's not easy for the developers responsible for the integration of these unfamiliar environments into the system. This is mostly because the structure keeps changing with each new announcement. The following may be missing that needs to be addressed,

- The background knowledge of the environment can be limited due to the non-existence of the community and proper documentation.
- The organizational approach towards problem-solving may differ.
- The lack of organizational or individual expertise in the implementation of modern methods.

However, there exist some more challenges as well. Each of the mentioned and these additional need to be addressed.

4. Importance

The development process is entirely changed in this rapidly evolving IT world where developers must continuously interact with new environments. This was not the case when the system used to be based on a few programming languages. But to survive in the competent software market, companies tend to use more advanced development methods where most can be achieved with limited resources including time and manpower [7]. And it's not straightforward to work in new environments every day. Therefore, the following are some significant factors in using environments effectively,

- Survival in a competent market is possible only with the use of the latest environments in the field of development.
- The continuity and system integrity can only be achieved with the effective use of these new methods.
- On the individual level, it gives credibility to the developer who is more exposed to working with the environment as it gives an extra advantage to resolve future problems in the relevant area based on previous experience.
- Any organization can reduce the resources with the use of effective troubleshooting methods and also win the confidence of other stakeholders.
- The overall system that uses the troubleshooting methods properly achieves business continuity and reduces downtime, ensuring the system's reliability.

5. Challenges

Restricted Recourses

Tools and dependencies are restricted during the initial stages of an environment. Sometimes limited access is provided to the developers due to the availability of the beta version of the system and not the final one. During this initial period, open-source contributions are also not possible, which in turn makes it more difficult to completely utilize the features of the system.

Limited Documentation

Most of the newly announced environments lack proper documents in the early stages of their announcement. This is because the community is mostly limited until the usage increases on a global level which is why there are limited people who are contributing towards the solutions to resolve complex problems while implementing the methods of the environment.

System Failure

There exists some risk of complete system failure if the environment is not used properly with the system. This can be due to missing constraints that are bound to be followed while doing integrations. On top of that, the initial procedures are not well equipped with the resolution methods that can be used to resolve the immediate failure. The troubleshooting methods can also be complex which makes the situation worse for the emergency team to recover the system.

Lack of Collaboration

In the initial days of the environment, there can be very limited sharing of knowledge as the community is more scattered. Most of the features are only used to a limited extent where developers are taking advantage of specific methods to resolve basic problems in their projects. Communication is also restricted due to complex initial terms used in the documentation which is more technical to understand in the initial period.

6. Current Troubleshooting Methods

The following are the already used troubleshooting methods being practiced in the software industry,

- Employing a reactive approach to resolve the conflict once occurred but it mostly fails in unfamiliar environments due to limited background knowledge.
- High dependence on documentation of the system but again this is not helpful for unfamiliar environments.
- Only relying on the experts of the existing systems who have limited knowledge of working in the new environments.
- Utilizing automated monitoring tools to identify the issues and resolve them automatically but the unfamiliar environments are accompanied by problems that are not addressed in the automated systems.

Therefore, more enhanced strategies are needed to overcome the challenges faced in exposure to the new environments.

7. Effective Strategies

The following problems exist in the current methods which are resolved with these suggested techniques,

- Lack of background knowledge
- Limited Collaboration
- Dependency on automated tools
- Inappropriate expertise
- Insufficient Documentations
- Lack of successful methods

So, consider the following strategies to address these issues,

Visual Aids

Use the visual aids in the following manner to better understand the problem and pave the way toward troubleshooting:

- Flowcharts and Sequence Diagrams to understand the complex architecture of the system and to point out the problems.
- Utilize tables to compare tools and resolution methods.
- Timelines to explore case studies and their effects.
- Other infographics during the process of conflict resolution.

Resource Utilization

Use advanced collaborative methods and respective tools that lead to the understanding of the system. Employ all possible existing methods that are helpful in the current implementation. Take help from the already available external resources like relevant documentation, and online surveys published by the relevant teams.

Expand Knowledge Circle

Initially start gaining knowledge with all the documentation available about the new system. Explore more with existing logs and other historical data that is relevant to the newly announced environment. Apart from the given documentation, make some extra effort with the use of visual tools to get familiar with the extended architecture of the system as well as the dependencies being used. Further, start contacting experts in the relevant field and gather all the possible knowledge and useful methods to fill the knowledge gap in the given documentation. In this way, most of the weak points can be avoided regarding the initial limited documentation of the environment. Use extended methods to collaborate effectively with the experts. If not directly associated, the related expertise can be explored, and peers can be contacted in an appropriate way which helps in solving complex problems [8].

Iterative Troubleshooting

During the initial integration, start with the simplest possible methods and test the results. Proceed only if there are no errors upfront and then keep making small changes in the given system with the use of environmental methods. Keep the previous iteration in the loop in case a problem is faced during the next one. Use iterative methods to resolve the problems faced during the integration. Keep the entire system safe by the small and manage changes. Verify and compare the progress with the previous cycle to keep going in the positive direction.

Prioritize the Tasks

Only integrate the mandatory functionalities to be added to the system. While troubleshooting the problems of these environments, the interest needs to be more inclined toward the most critical tasks to avoid potential failure.

By employing the prioritization of tasks, one can therefore avoid the risk of complete system failure. A comprehensive plan, however, is required to carry out the priority steps during the integration of the new features.

Advanced Diagnostic Methods

Use advanced methods like Root Cause Analysis (RCA) to find the base of the problem and avoid conflicts. Keep questioning until the problem is explored completely and the root cause is found [9], [10]. Use the diagrams to visualize the causes of the problem instead of relying on the limited resolutions of the system. Moreover, employ modern technologies and algorithms to automate the basic steps and shift the focus toward more important activities in the process.

8. Best Practices

The following are some recommendations for working with unfamiliar environments to troubleshoot with fewer problems,

- Keep in touch with the latest methods to utilize modern tools and avoid complex implementations.
- Completely understand the documentation beforehand to save the extra resources in later stages.
- Be clear while communicating the requirements between the responsible team and other stakeholders involved.
- It's mandatory to Document everything while working in unfamiliar environments to leave proper footprints. In this way, the future members do not have to start over.
- Use feedback methods to take a look at the past and notice the success and failure ratio during the process.

9. Future Development

More advanced methods are predicted to be announced considering the ongoing research that will make the life of developers easy. The state-of-the-art machine learning algorithms will help achieve automation and implicit methods associated with the environment. For instance, context-aware methods and automated diagnostic methods are continuously evolving with the help of AI. Other than that, advanced collaborative methods are being developed which helps maintain the connection between the developers and also retain the documentation of their implementation methods. The fault tolerance is continuously increasing the systems in addition to the security integrations with legal considerations.

10. Conclusion

As a cessation, the troubleshooting methods for unfamiliar environments can be complex sometimes and therefore require complete theoretical and prior experience of working in the relevant environments. Other soft skills are also required for better communication between the stakeholders of the project. Having all these strategies in hand, it becomes easier to work with the new environments. The best practices suggested if following in addition to the prior strategies make it easy to achieve optimized and faster results.

With the continuous development and evolution in technology, the implementation and handling of unfamiliar environments will be comparatively simple. More optimized and easy methods are being developed which helps effectively solve the implementation problems. The confidence interval is therefore going to be more precise.

References

[1]. H. Gredler, "Troubleshooting," The Complete IS-IS Routing Protocol, London, 2005.

[2]. K. Wang, C. Fung, C. Ding, P. Pei, S. Huang and Z. Lua, "A methodology for root-cause analysis in component-based systems," in IEEE 23rd International Symposium on Quality of Service (IWQoS), Portland, OR, USA, 11 Feb 2016.

[3]. M. Tejeiro Koller, "Exploring adaptability in organizations: Where adaptive advantage comes from and what it is based upon," Journal of Organizational Change Management, vol. 29, no. 9, pp. 837-854., 03 Oct 2016.

[4]. A. Colman and J. Han, "Using role-based coordination to achieve software adaptability," Science of Computer Programming, vol. 64, no. 2, pp. 223-245, 15 Jan 2007.

[5]. D. Sotirovski, "Heuristics for iterative software development," IEEE Software, vol. 18, no. 3, pp. 66-73, 07 Aug 2002.

[6]. B.-Y. Tsai, S. Stobart, N. Parrington and B. Thompson, "Iterative design and testing within the software development life cycle," Software Quality Journal, vol. 6, pp. 295-310, Dec 1997.
[7]. B. Dagenais, H. Ossher, R. K. E. Bellamy, M. P. Robillard and J. P. d. Vries, "Moving into a new software project landscape," ICSE 10: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 1, pp. 275-284, 01 May 2010.
[8]. K. O. James A. Highsmith III, Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, New York: Dorset House Publishing.
[9]. P. M. Williams, "Techniques for Root Cause Analysis," Baylor University Medical Center Proceedings, vol. 14, no. 2, pp. 154-157, 2001.
[10]. M. Solé, V. Muntés-Mulero, A. I. Rana and G. Estrada, "Survey on Models and Techniques for Root-Cause Analysis," Cornell University, 03 Jul 2017.