ISBN: 978-989-8533-57-9 © 2016

STRATEGIES FOR ZERO-DOWNTIME RELEASES: A COMPARATIVE STUDY

Diego Quirino Silva, Everton Gomede and Rodolfo Miranda Barros
Universidade Estadual de Londrina
Londrina, Parana, Brazil

ABSTRACT

Adopting continuous delivery means facing new challenges: it requires, as one of the conditions, to manage continuous deployment of new versions or bug corrections. However, lack of planning may cause the system or service to be temporarily unavailable, and because of that approaches on how to decrease risks (or manage zero risk) are needed. This study presents a comparative analysis on the advantages and disadvantages of possible strategies, as well as a description of an approach believed to solve the unavailability problem.

KEYWORDS

Zero downtime, continuous delivery, virtualization

1. INTRODUCTION

The software development process includes several stages. The most critical is the delivery stage, from which the software truly begins to deliver value for the client.

However, by adopting the continuous delivery approach, which is the ability of delivering reliable and quality software at any time (Humble and Farley 2010), new challenges arise. One of them is related to the moment of providing users with the new application or update. This is usually a moment when errors can be made (even assuming tests were performed in previous steps) causing the application to be unavailable for minutes or even hours (worst scenario).

In a continuous delivery environment (Humble and Farley 2010) in which the application can be deployed several times a day, it would be acceptable to risk having it unavailable for the same number of times.

Such unavailability is known as downtime. An unavailable system or service, for whatever the reason, will impact on the corporation and its clients: in risk operation environments this can only be acceptable under SLA (Humble and Farley 2010), and it could still cause losses (Oracle 2015; Silva et al. 2009). According to (Humble and Farley 2010), zero downtime in continuous delivery is the ability to drive users between versions in an almost instantaneous way.

This study aims to present a comparative analysis on the strategies of migration downtime in the shortest possible time, or time equals zero. The transition from the previous to the latest version must be managed with minimal impact, and must be imperceptible to users, because when they face service unavailability the trust rating decreases ultimately resulting on giving up using the system.

The means to solve this problem include mitigation or elimination – the possible strategies are: blue-green deployment, canary release or, in last case, rollback (Humble and Farley 2010). Adopting one or more strategies may provide the corporations with the essential competitive differentiation.

This study is divided as: in Section 2 a theoretical foundation takes place in order to present the concepts; in Section 3 the approaches and possible strategies are described; in Section 4 the results of the comparative analysis and the strategies limitations are reported; and lastly, in Section 5, the conclusion and possibilities for future studies are presented.

2. THEORETICAL FOUNDATION AND RELATED STUDIES

Continuous delivery is the ability to create value from an internal or external idea, which means, in other words, having the software working in the shortest possible time (Akerele et al. 2013).

Continuous delivery brings many visible and tangible benefits. Authors (Neely and Stolt 2013; Chen 2015) stress that even in corporations working with an adequate software process structure, continuous delivery allows improvements.

Achieving continuous delivery demands more than the adoption of suggested practices and techniques – it requires an effort towards architecture planning and projecting, as well as understanding related costs. Once the project is defined, the environment becomes ready to continuous deployment, which must be a quality attribute (Bellomo et al 2014). Defining tactics in order to achieve this attribute is supported by (Bellomo et al 2014), who stresses that tactics must allow continuous integration, enable test automation, enable fast and solid operation deployment, and allow flexible and syncable environments.

Downtime sources: lack of control over changes in the production environment (Humble and Farley 2010), software aging causing degradation of the application over time, resulting in unavailability (Silva et al. 2009; Thein et al. 2008; Shetty et al. 2008).

Regarding the cons of downtime, a qualitative research conducted by (Oracle 2015) shares the following findings: 50% impact in interruption of IT businesses and services, 37% decrease in productivity, 34% negative impact on IT corporations, and 24% negative impact on businesses.

Hardware and/or software replication is one of the techniques applied in attempting to avoid downtime in high-availability environments; the way it is performed vary among the authors (Gavrilovska et al. 2002; Silva et al 2009; Thein et al. 2008; Shetty et al. 2008), but replication is always stated as the chosen method.

While choosing the strategy to mitigate downtime, it is necessary to consider the TCO (Total Cost Ownership) in order to define the best solution. This scale measures all the costs involved in software resources and the corporative infrastructure, from purchasing to disposal.

MTTR (Mean Time to Repair) is another indicator related to the time spent in order to repair something – an error or service unavailability. It is obtained by the total duration of failures divided by the total number of failures. E.g. MTTR = 180 min / 6 failures = 30 minutes (Agarwal et al. 2010).

The SLA (Service Level Agreement) is used to determine (in a contract) the level of obligation by which a service should remain available. It is mostly used in cloud computing, but it is not restricted to that, also being associated with telecommunication services, web services, or any other in which the client requires functioning assurance in a specified period of time (Stamou et al. 2013; Shu and Meina 2010).

TCO, MTTR, and SLA are extremely important because they allow the selection of the strategy to be adopted, as well as verify whether the results achieved meet what was initially planned.

2.1 ITIL

In order to be successful, corporations must make efforts and provide resources towards optimization. However, this is not a reality in all corporations, especially regarding the IT – which is underestimated and misunderstood, considered to be expensive and not a profitable activity.

This is due to the apparent detachment between IT and corporative goals, because many times the IT is not able to properly communicate the benefits of adopting a specific technology or the implementation of a new approach for given problems or market opportunities. This gives space for misunderstanding, or lack of understanding of those charged with the responsibility to carry out new plans.

In this context, the OGC (Office of Government Commerce) developed, in the 1980s, the first version of ITIL, which continued to be improved in the next few years, coming to its latest version (3) in 2007 (Esmaili et al. 2010).

ITIL (Information Technology Infrastructure Library) is a framework consisting of a collection of guidelines to design, implement, and maintain IT services. From the third version on, ITIL included 5 books that are core and cover the service lifecycle from beginning to end (Esmaili et al. 2010). The books are divided as: service strategy, service design, service transition, service operation, and continual service improvement.

ISBN: 978-989-8533-57-9 © 2016

The implementation of ITIL in a corporation allows the alignment of IT and the corporative strategic objectives (Esmaili et al. 2010), supporting high-level performance. Nevertheless, it is not an easy process, because even though ITIL provides the corporation with suitable guidelines, it does not present practical solutions – that being responsibility of each corporation.

2.1.1 Service Transition

From all ITIL framework books, service transition is the most suitable for the understanding of this study. Corporations seek to meet market requirements, which means transforming opportunities in profitable activities, while keeping regular users safe. Thus, it is important to define the basic procedures so that both opportunity and risk decrease come safely and effectively.

The service transition process accurately covers the aspects of status change from development to delivery (Eikebrokk and Iden 2012). This is important because most of the software products are considered to be services – some are essential – and they cannot stop working. Besides that, the service transition combines elements such as the release management and the project risk management, placing them in the context of service management.

With continuous delivery, improvement flows continuously as well as the transition between service versions, and adopting processes to make this technique successful becomes a necessity.

The basic processes for service transition, as described in ITIL framework are:

Transition planning and support: this process aims to plan and coordinate resources to assure the new function will meet what was pre-established in terms of cost, quality, and time; and ensure standardization of a reusable framework and provide comprehensible planning in order to clarify the plans of service transition (Service transition 2007).

Change management: aims to ensure the changes will be recorded and validated, authorized, prioritized, planned, tested, implemented, documented, and reviewed in order to be controlled (Service transition 2007).

Service asset and configuration management: defines and controls infrastructure service components and properly maintains configuration information (Service transition 2007).

Release and deployment management: allows the planning of deployment and release to be easily understood so that both clients and businesses can be aware of such changes; successfully reaches build, installation, and testing in the target environment according to the deadline; assures the deployed change is able to meet agreed requirements; assures minimal impact in actions of service, operation and support to the corporation; and assures clients, users, and the service management team are satisfied with the service transition (Service transition 2007).

Service validation and testing: aims to assure the release process will be trustful in the sense that new services reach results that meet the desired cost, capacity, and restriction values; aims to guarantee the delivered service meets expected performance levels; and to confirm the defined requirements of users and stakeholders are met, in order to avoid having corrections on deployed services (Service transition 2007).

Evaluation: aims to guarantee the intended changes in the service will meet agreed levels of corporative capacity, resources, and restrictions provided; provides good outcomes so that the change management can decide to approve and start performing the change (Service transition 2007).

3. ZERO-DOWNTIME STRATEGIES

For many corporations, the software is more than a service – it is the very condition of their existence, for instance: Netflix®, Google Doc®, and Home Broker of financial institutions. These corporations cannot face unavailability, because having a service interrupted can cause the satisfaction level of clients to decrease, as well as result in harmful financial trials.

As long as the failures take place in controlled environments, such as development or homologation, it is not a matter of concern. However, when these affect the client they must be reduced or eliminated, because it may decrease the user trust level.

Most of users hardly take service unavailability easily, because in same cases the services are fundamental, such as in financial or health care situations.

In order to mitigate or eliminate the service unavailability there are some strategies, such as the blue-green deployment and the canary release.

3.1 Blue-green Deployment

The blue-green strategy works with the commutation concept between production environments, namely blue and green, being that the new version is deployed in one of these environments, and the users' requests are redirected to them (Humble and Farley 2010), as shown in Figure 1.

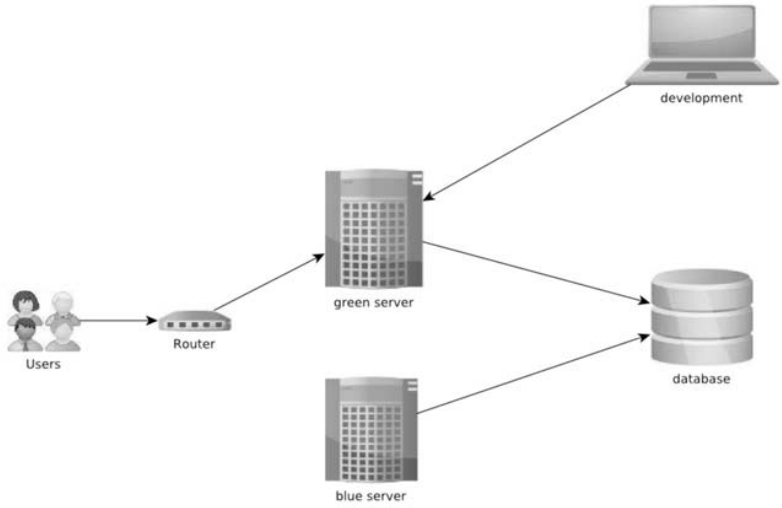

Figure 1. Blue-green deployment

During the deployment of the new version, failures do not affect clients, who will be in a different environment and protected from any side effect of the new version. Thus, the service is not interrupted, allowing the error source to be detected and corrected.

If the deployment happens as expected and does not face any obstacle, the clients are commuted to the new version through the router configuration, being able to start using it.

It is true that some errors can only be perceived in cases of system overload. In that case, if any service instability or degradation is reported after the clients' commutation to the blue environment, the switching process should be reverted to the previous version, avoiding the complete unavailability of the application.

Even in corporations with few resources this can be performed simply by setting the application to run (at the same time and independently) in the same server, using virtualization technologies, with no need for a real distribution environment. Corporations enjoying more resources may use the same strategy, adding a real distributed server system.

It is important to mention, also, that the database used by the application may be the source of the service interruption or instability, meaning that precautions must be taken. Firstly, the reason for the new deployment must be defined, whether it is an update of the application or data change.

The application update – assuming the database would be the same for any version of the running application – will not affect the clients. However, in case of changes in the data scheme, they should be performed prior to the application interaction starts, through refactoring the database that supports both previous and new versions (Fowler 2010).

3.2 Canary Release

Canary release is a technique by which only a small pre-selected group of users is granted access to the new version, as seen in Figure 2, minimizing deployment-associated risks (Humble and Farley 2010; Sato 2014).

ISBN: 978-989-8533-57-9 © 2016

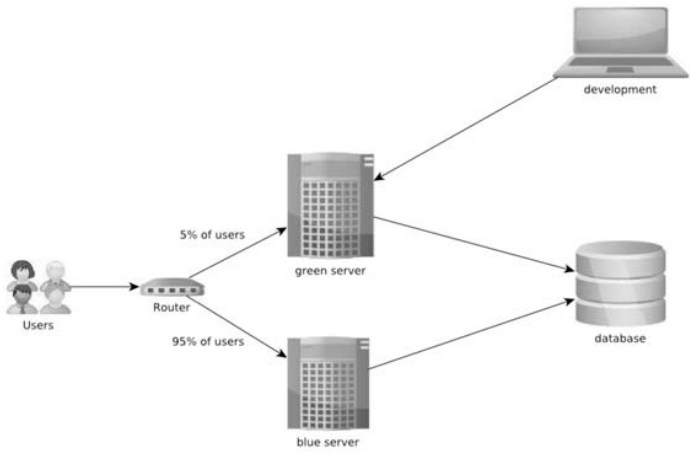

Figure 2. Canary release

Having only a single version of the application brings benefits such as better bug correction management and infrastructure, but it may also have side effects. This is true in the sense that bugs would only show up while the application runs in the production environment (Humble and Farley 2010).

This technique allows zero downtime by subjecting only a small group of users to possible errors.

The benefit of canary release is that at any sign of problem – related to inadequate functioning of a new application functionality, or below expected performance – it will only affect a small and more tolerant group of users, which differs from the aforementioned strategy where all users could suffer. It also gives the possibility to rate the most used functions and discontinue the ones that are not being used (this technique is known as A/B test). This technique keeps developers from performing useless tasks, allowing them to work with more important functionalities (Humble and Farley 2010).

This strategy requires much more hard working with the chosen architecture, because it demands specific configurations so that only a small group of users is granted access to the new functionality.

3.3 Lightweight Virtualization

The aforementioned strategies require means of performing deployment, and one of the possibilities is through virtualization. Virtualization is the technology used in the main cloud computing services, for it allows the best use of available resources. A new virtualization method, which is important to mention, is known as lightweight virtualization.

This technology was developed to work as a type of virtualization, but focused on smoothness and speed. In the same sense of conventional virtualization, it tends to have a lower TCO for the users. The major difference between the techniques is that lightweight virtualization does not use virtual machine monitor, redundant operating system kernel, and libraries – all of which makes it lighter (Morabito and Komu 2015).

The container technology became popular with the development of Linux special versions. A container is a repository of several resources, such as CPU, network input and output, and host machine RAM memory. Some of the products available are: LxC, Docker, Linux V-Server, among others (Morabito and Komu 2015).

The technology also includes some application models, some of which are specifically designed for delivery strategies, having an interface that communicates with the containers in a service level. The containers' ability to isolate from others helps testing and development, and can be used jointly with the continuous integration process (Morabito and Komu 2015).

Although there are advantages, the technology still faces some limitations, as presented by (Morabito and Komu 2015), such as lack of interoperability in cases of difference between the container operating system and the host operating system; and safety issues, given that the kernel system is shared by all containers.

4. COMPARATIVE ANALYSIS DISCUSSION

The possible approaches for downtime mitigation may include the following strategies: Blue-green and lightweight virtualization (Morabito and Komu 2015), Blue-green and virtualization (Silva et al. 2009; Thein et al. 2008; Shetty et al. 2008), Canary and lightweight virtualization, and Canary and virtualization. Both the blue-green and the canary release strategies may have to deal with issues of distributed infrastructure, which would demand from the corporation (aiming to improve its availability) to increase resource investment, and this may not always be the case.

Corporations do not usually have idle servers (because of the cost) to apply a distributed blue-green strategy, and on the other hand there may be servers unable to operate with all the available hardware resources. By using lightweight virtualization or virtualization technologies both problems could be avoided at once.

Virtualization is a tested and supported technique associated with environments hosting high-availability services. Virtualization provides capacity, for instance, for a server to create several virtual resources (Thein et al. 2008; Sagana 2013), allowing the corporation to perform tasks in a single server. E.g. running several levels of a web server, each having their own networking, but sharing the server's RAM memory (Vaughan-Nichols 2006).

Blue-Green and lightweight virtualization

Advantages: low cost – does not require new equipment purchase; better performance – considering the lightweight virtualization technology works with less software layers, it achieves better processing (Vaughan-Nichols 2006).

Disadvantages: high complexity for installation and management of the lightweight virtualization, because it is a new technology (Vaughan-Nichols 2006); requires specialized labor – deployment and management of virtualized environments require high-level knowledge; and it may cause impact on some users in case the new deploy presents problems regarding new functionalities (not included on previous versions).

Blue-Green and virtualization

Advantages: easy to implement – it only requires the replication of the automation and production environments in order to commute them to each new deploy, or defining permissions so that someone can be in charge of performing the tasks; low cost – does not require new equipment purchase; lower installation complexity compared to lightweight virtualization.

Disadvantages: requires specialized labor – deployment and management of virtualized environments require high-level knowledge; and it may cause impact on some users in case the new deploy presents problems.

Canary and lightweight virtualization

Advantages: low impact on users, because only a small and more tolerant group of users will be affected; low cost – does not require new equipment purchase.

Disadvantages: requires specialized labor – deployment and management of virtualized environments require high-level knowledge; high complexity – requires configuration of both the virtualization technology and the network, in order to select the users who will have access to the new version.

Canary and virtualization

Advantages: low impact on users, because only a small and more tolerant group of users will be affected; low cost – does not require new equipment purchase; better performance – the container technology operates with a layer less of software.

Disadvantages: average complexity – requires higher network configuration so that only a restricted group of users has access to the new version.

Table 1 and Table 2 present the advantages and disadvantages of each of the combinations described. A more detailed description of the tables is presented below.

ISBN: 978-989-8533-57-9 © 2016

Table 1. Combined approaches advantages

	Blue-green & Virtualization	Bue-green & Lightweight Virtualization	Canary & Virtualization	Canary & Lightweight Virtualization
Low cost	●	●	●	●
Clear for the user			●	●
Low complexity	●			
Performance		●		●

Table 2. Combined approaches disadvantages

	Blue-green & Virtualization	Bue-green & Lightweight Virtualization	Canary & Virtualization	Canary & Lightweight Virtualization
Specialized labor	●	●	●	●
High complexity		●	●	●
Impact on users	●	●		●

Tables 1 and 2 detailed description:

Advantages:

1. Low cost – None of the combinations require new equipment purchase;
1. Better performance in Blue-green/Canary & Lightweigh Virtualization cases, because the container technology works with a layer less of software and has better processing performance (Vaughan-Nichols 2006) ;
2. Easy implementation in Blue-green/Canary & Virtualization – it only requires the replication of automation and production environments in order to commute each new deploy;
3. Low complexity of installation in Blue-green/Canary & Virtualization compared to lightweight. The lightweight virtualization technology is more recent (Vaughan-Nichols 2006) when compared to the standard virtualization;
4. Low impact on users in Canary & Virtualization/Lightweight Virtualization – only a small and more tolerant group of users will be affected;

Disadvantages:

1. High complexity in Blue-green/Canary & Lightweight Virtualization, because the lightweight virtualization technology is more recent (Vaughan-Nichols 2006);
2. Requires specialized labor – all the cases require high-level knowledge in order to use virtualization;
3. Blue-green & Virtualization/Lightweight Virtualization – can impact on some users in case the new deploy presents errors regarding new functionalities (not included on previous versions);
4. High complexity in Canary & Virtualization/Lightweight Virtualization – it requires higher network configuration so that only a small group of users is granted access, besides the complexity of lightweight virtualization itself.

Such approaches are – theoretically – efficient because the virtualization brings several benefits and can be combined with the last stage of continuous delivery.

Figure 3 shows the steps after the new functionality commits in the control system of the version, using one of the possible approaches:

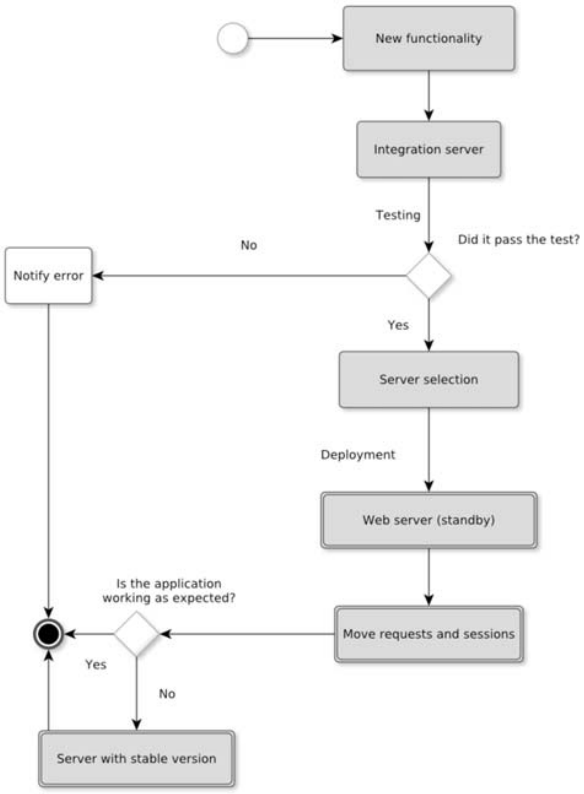

Figure 3. Activity diagram

The virtualization approach described by (Silva et al. 2009) brings solid results and the description matches with the blue-green strategy – the benefits include achieving goals and objectives.

The virtualized environment described by (Silva et al. 2009) consists of three environments:

The first virtualized resource is the load balancer, which is responsible for detecting performance problems and errors that could only be perceived in a production environment with full access by the users – problems that could not be detected in previous testing or in the homologation environment. This component is responsible for moving the clients from one environment to another in the described situations, but in order to answer continuous delivery effectively it must also work such that when a new deployment occurs, the clients would be moved automatically to the new version, and not just in case of failures.

The second and third virtualized resources would lie inside the production environment itself, playing the role of blue and green environments. These components are where the application server is installed. Because it is a virtualized environment, servers from any producers can be adopted.

This approach brings several benefits, such as:

The virtualized resources could be described in script language in order to run through automatized processes – as suggested by (Humble and Farley 2010) – and, thus, work together with the version control system.

Given that the environment is a result of virtualization, there would be no need to purchase new hardware, subsequently decreasing the TCO.

The environment would be properly instrumentalized – as described by (Silva et al. 2009). Once the software stopped meeting defined performance requirements or detected log failures, the clients would be moved back to the previous steady version.

This approach is supported by the results shown in authors' (Silva et al. 2009; Thein et al. 2008; Shetty et al. 2008) studies, which demonstrate a virtualized environment that provides high-level availability, and even

though they focus on the software aging issue, it can be applied to continuous delivery and adapted to the blue-green strategy. Table 3 presents criteria for selecting this approach.

All achieved goals are aligned to unavailability risk minimization and can be adapted to the blue-green strategy.

Table 3. Related studies comparative of virtualization use

Criteria	(Silva et al. 2009)	(Thein et al. 2008)	(Shetty et al. 2008)
Self-recovery	●	●	●
Applicable on any server	●		
MTTR decrease	●		
Virtualization	●	●	●
No loss of requests or sessions	●	●	
Manageable in just one computer	●	●	●

The last topic to stress is that the combined approaches described previously present adhesion points between them and the ITIL framework – especially the service transition steps. The adhesion topics are shown in Table 4:

Table 4. Adhesion between trasition service and aprroaches

Processes	Blue-Green/Canary & Virtualization/Lightweight virtualization
Transition planning and support	
Change management	
Service asset and configuration management	●
Release and deployment management	●
Service validation and testing	●
Evaluation	●

The adhesion between framework and approaches is described as follows:

Service asset and configuration management: according to (Humble and Farley 2010), when it comes to virtualized environments, all the configurations and specifications of the used environments can be described in script language, and stored in version control systems, so that every and any modification will be recorded – and can be undone. The modifications will also be properly documented.

Release and deployment management: in virtualized environments, release and deployment tasks can be performed by automatized services, with the only need to define permission preferences to select who will be able to perform these tasks and who will not.

Service validation and testing: all the testing and validation steps are clearly defined in the continuous delivery process described by (Humble and Farley 2010) in the pipeline. Prior to arriving at the release and deployment stages, several tests are performed and only if they are successful the service can move to the next stages.

Evaluation: after the deployment in the selected environment, stress tests can be performed in order to confirm whether it meets pre-defined requirements – only if it passes the tests the commutation between virtualized environments is performed. In more specific situations, the commutation stage can be performed manually, but only after the responsible for release approves it.

Processes not adherent to approaches:

Transition planning and support, and change management: These are not included in the approaches because they escape their scope – both processes focus on activities that are prior to the deployment of a new service.

This approach is more suitable to corporations with available equipment and at least a few initial resources, but it is not restricted to them.

4.1 Limitations

There are some limitations and disadvantages related to the proposed approach, which needs proper attention.

First, even tough it does not require new material resources, it demands human resources with high-level knowledge in virtualization, so that it can work adequately and as expected.

Second, if the clients using new functionalities are moved to the previous version due to some error, they may face unavailability, because the new functionality is not included in the previous version. Therefore, the session or request will not be re-established, causing them to lose in-progress work.

Third, failures occurring in the single server with virtualized environments will cause service unavailability anyways.

Forth, the issues related to the database are not mentioned in the described approaches. Thus, this is a question yet to be solved in this context.

5. CONCLUSION AND FUTURE STUDIES

This study attempted to provide a comparative analysis between the possible solutions related to the unavailability problem, as well as suggest means of dealing with downtime – we believe the described technique is a possible path to follow. In general, the described approaches succeed in helping corporations with few available resources meet their clients' expectations. Finally, it is important to mention this is a theoretical study that needs to be tested in order to provide solid results for possible users.

REFERENCES

Agarwal, B., Tayal, S., and Gupta, M. (2010). *Software Engineering and Testing*. Computer science series. Jones & Bartlett Learning. Pages: 110-112.

Akerele, O., Ramachandran, M., and Dixon, M. (2013). System dynamics modeling of agile continuous delivery process. *In Proceedings - AGILE 2013.*

Bellomo, S., Ernst, N., Nord, R., and Kazman, R. (2014). Toward design decisions to enable deployability: Empirical study of three projects reaching for the continuous delivery holy grail. *In Proceedings of the International Conference on Dependable Systems and Networks*.

Chen, L. (2015). Continuous Delivery: Huge Benefits, but Challenges Too. *IEEE Software,* 32(2):50–54.

Eikebrokk, T. R. and Iden, J. (2012). ITIL Implementation: The Role of ITIL Software and Project Quality. *In 2012 23rd International Workshop on Database and Expert Systems Applications* , number January, pages 60–64. IEEE.

Fowler, M. (2010). Blue-green deployment. Available at: http://martinfowler.com/bliki/BlueGreenDeployment.html. (Accessed on 06/28/2016).

Gavrilovska, A., Schwan, K., and Oleson, V. (2002). A practical approach for 'zero' downtime in an operational information system. *In Proceedings 22nd International Conference on Distributed Computing Systems*, pages 345–352. EEE Comput. Soc.

Esmaili, H. B., Gardesh, H., and Sikari, S. S. (2010). Strategic alignment: ITIL perspective. *In 2010 2nd International Conference on Computer Technology and Development*, number Icctd, pages 550–555. IEEE.

Humble, J. and Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison Wesley Signature Series (Fowler). Pearson Education.

Morabito, R., Kjallman, J., and Komu, M. (2015). Hypervisors vs. Lightweight Virtualization: A Performance Comparison. In 2015 IEEE International Conference on Cloud Engineering , pages 386–393. IEEE.

Neely, S. and Stolt, S. (2013). Continuous delivery? Easy! Just change everything (well, maybe it is not that easy). I*n Proceedings - AGILE 2013.*

Oracle (2015). Zero downtime database upgrade using oracle goldengate. Available at: <http://www.oracle.com/technetwork/middleware/goldengate/overview/ggzerodowntimedatabaseupgrades-174928.pdf> (Accessed on 06/28/2016).

Sagana, C., Geetha, M., and Suganthe, R. C. (2013). Performance enhancement in live migration for cloud computing environments. *In 2013 International Conference on Information Communication and Embedded Systems (ICICES)*, pages 361–366. IEEE.

Sato, D. (2014). Canary release. Available at: http://martinfowler.com/bliki/CanaryRelease.html. (Accessed on 06/28/2016).

Service transition (2007). *Service transition IT infrastructure library*. Stationery Office, 2007. pages 42-84.

Shetty, H., Nambiar, M., and Kalita, H. (2008). Analysis and application of conditional software rejuvenation — A new approach. *In 2008 IEEE International Conference on Software Reliability Engineering Workshops (ISSRE Wksp)*, pages 1–5. IEEE.

Shu, Z. and Meina, S. (2010). An architecture design of life cycle based sla management. *In Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, volume 2, pages 1351–1355.

Silva, L. M., Alonso, J., and Torres, J. (2009). Using Virtualization to Improve Software Rejuvenation. *IEEE Transactions on Computers*, 58(11):1525–1538.

Stamou, K., Kantere, V., and Morin, J.-h. (2013). SLA data management criteria. *In 2013 IEEE International Conference on Big Data*, pages 34–42. IEEE.

Thein, T., Chi, S.-d., and Park, J. S. (2008). Improving Fault Tolerance by Virtualization and Software Rejuvenation. *In 2008 Second Asia International Conference on Modelling & Simulation (AMS)*, pages 855–860. IEEE.

Vaughan-Nichols, S. (2006). New Approach to Virtualization Is a Lightweight. *Computer*, 39(11):12–14.