IMPACT: International Journal of Research in Engineering & Technology (IMPACT: IJRET)
ISSN (E): 2321-8843; ISSN (P): 2347-4599
Vol. 4, Issue 5, May 2016, 77-98

INNOVATIVE SOFTWARE DELIVERY FRAMEWORK TO WARDS SOFTWARE APPLICATIONS MODERNIZATION

SAMER I. MOHAMED

Modern Science and Arts University, Faculty of Engineering, Electrical and Communication Department, Egypt

ABSTRACT

Many of the current IT organizations are shifting their business and service offerings to satisfy and meet their clients demand via new approaches like application modernization. Software modernization is refactoring or consolidation of the traditional and legacy software engineering/programming techniques to be aligned with the business outcomes and client needs. Operational excellence, service alignment and portfolio optimizations are the key strategies adopted by the IT organizations to tailor their IT services/solutions to meet their client business objectives. This article presents an innovative E2E (End-to-End) framework that articulates the next generation of software application development and delivery. The proposed framework is designed to bridge the current gap with the legacy approaches through linking together available resources, tools, infrastructure, information services, and delivery approaches in seamless and lean manner to maximize the business value towards end users. The edge and uniqueness of this framework is that it focuses in aligning the software service offerings with the business outcomes via E2E approach starting from assessing the client needs and draft a plan of actions to transform into the new way of IT delivery.

KEYWORDS: Cloud Computing, Application Modernization, Operational Excellence, Software Refactoring, Application Transformation to Cloud

1. INTRODUCTION

Software modernization is a way of shifting the legacy and traditional approaches of software applications development and delivery to cope with the new trends of IT and being in sync with the current and future approaches market needs. This puts high pressure on Software and IT organizations to adopt a new way to make their software products and services able to survive in the current competing market. Thus IT organizations starts to shift their service offering and software application delivery to adopt new emerging approaches/strategies to cover these needs and satisfy their client's demands. Shifting IT towards a new era of cloud computing becomes the focus and main interest for most of the current clients and organizations because it radically changes the way infrastructure, applications, and information are sourced and consumed through highly scalable and elastic technology-enabled services on a pay-per-use basis. This paper highlights the big challenges for transforming not only the current software applications development and delivery approaches and practices, but also how the organizations to play according to the new rules of the IT services from application delivery perspective. It also drafts a new and simple delivery transformation model helping achieving this objective [5].

IT has always been in the forefront of innovation and agility. New applications fuel an organization's growth and productivity, automate key business processes and provide competitive edge to survive with current competition. However,

they are often written in outdated development languages and run on platforms that can no longer effectively integrate with the organization's new architectures. That is why many of current IT organizations adopted new strategies to transform their legacy and traditional way of software development and delivery through new approaches like application modernization, application transformation to cloud, and new style of delivering IT services. Linking all this together helps the organizations to overcome and mitigate current risks/challenges of application growth, maintenance cost, and provide real business values to their clients and end users.

Application transformation and modernizations does not come easy. Breaking existing patterns requires a series of bold, and often, disruptive steps. It is simply not realistic to wipe the slate clean and start building the new application landscape from scratch. Instead, IT teams need to find the most impactful solution and design a gradual approach to restructuring and reorganizing their portfolios. With this in mind, adopting the new style of software development and delivery bring significant benefits like:

- Better aligned application landscape
- Reduced IT operating costs
- Improved agility of existing applications
- Tighter alignment with the business
- Better compliance with data retention regulations and easier access to archived records
- Improved processes going forward to prevent future problems
- Renewed focus on innovation

The paper is organized as follows: section II gives a background for the early studies done on software modernization and new style of IT services as a concept along with an overview of the challenges faces by the current organizations; section III illustrates the proposed E2E framework and how IT services can be delivered in seamless strategy under proposed framework. Section IV introduces the proposed Proof of Concept (PoC) model; where PoC description, details, results, and recommendations are detailed; section V is the conclusion of this study.

2. NEW STYLE OF SERVICES & SOFTWARE MODERNIZATION BACKGROUND

The New Style of Services is where cloud, security, big data, and mobility all converge in comprehensive solutions to better connect customers, citizens, communities, partners, and suppliers to your client's enterprise [16]. The New Style of IT services supports enterprise goals as it gives clients choice as to how to consume IT services, where to get it, and how to pay for it. It is efficient, open, flexible, scalable, and collaborative. It will predominantly be made up of as-a-Service (aaS) and consumption-based models. The new style of IT services raises the user's expectations of instant connection to information any time and from anywhere. It is driven by business outcomes to satisfy the organization goals. This new world where cloud, security and big data all converge to better connect with customers, partners and suppliers To stay relevant, enterprises need infrastructure, services, and software systems that can drive connections, generate business insights, and help the organization succeed and grow. Software delivery for Cloud is one of the new approaches to delivery IT services and it is designed to help organizations seamlessly and securely manage, operate, produce, maintain, and

enhance their software applications and IT services via a dynamic approach that provides the reliability, predictability, and efficiency they need to ultimately get the most from the applications portfolio. Adopting the New Style of IT services best practices enable organizations to:

- Create new outcomes for apps, data and business processes/IT experiences.
- Compose new apps and services from available sources to create new value for the customers.
- Manage proactively all forms of risks with world of rapidly changing threats.
- Develop real time instant insights for continuous improvements, innovation and learning.
- Satisfy customers' needs and cope with market dynamics.
- Support growth strategy.
- Help employees collaborate better and balance work/life
- Bridge current challenges of traditional and legacy IT services delivery.
- Deliver in Smarter rather than harder approach.
- Support operational excellence via industrialized delivery model to sustain quality while reducing cost.
- Maintain innovative approach to align services to the business.
- Portfolio optimization - Techniques and tools to systematically balance the portfolio
- Measure, monitor, track, report and continuously improve on its contribution to the business success.
- Provide the reliability, predictability, and efficiency needed to ultimately get the most from the applications portfolio.
- Provide management expertise across diverse application technologies delivers benefits and best practices of world-class applications management services adapted to the unique needs and increased flexibility of a cloud environment.
- Enable organizations to focus on innovative/value add projects by providing effective and efficient support across a hybrid environment.

Figure 1: New Style of IT Services Objective

There are many challenges faced by the organizations to deliver as per the New style of IT services best practices and get benefit from the cloud computing edges, to satisfy their clients' needs on different aspects like security, open stack delivery, open standards and COTS support. This can be summarized as follows:

- Security
  - Security and data privacy needs to be at the center of business solution design – rather than an after-thought.
- Data growth
  - Social computing applications, big data, and rich media provide greater insight into customer preferences and need to be analyzed and action taken.
- Cloud and mobile devices growth
  - Cloud-based applications now challenge the status and how IT services are delivered.

Mobile applications and increasingly sophisticated smart phones and tablets provide anywhere, anytime access and create new challenges for the enterprise when employees prefer to use them at work.

3. PROPOSED TRANSFORMATION AND MODERNIZATION FRAMEWORK

The proposed E2E framework is introduced to tackle the challenges with the current legacy and traditional approaches of software development and delivery faced by current IT organizations like mentioned in the previous sections. The uniqueness of the proposed framework can be summarized as follows:

- Innovative approach to align services to the business outcomes.
- Narrow down overhead communication and build on collaboration.
- Provides reliability, predictability, and efficiency to ultimately get the most from the applications portfolio.
- Assess project maturity through innovative maturity calculator tool.
- Utilize outcomes from maturity calculator to draft action plan via CSFs/KPIs.

- Automated environment setup toolkit based on push button approach.
- Facilitate smooth/seamless delivery model with all interlocked teams.
- Plugin/customize the tools/resources to fit for project purpose
- Adapt with the new IT trends and converged infrastructure, information, services and delivery approaches.
- Enable services flexibility and portability.
- Articulate seamless and lean delivery approaches.
- Utilize available resources/tools to maximize value towards clients.
- Sustain quality while reducing cost via industrialized delivery model.

The framework consists of three main components:

- Maturity calculator
- Push-button Environment Setup toolkit
- Delivery model

Through the next sections, I will go through the details for each component, how it works, value behind and how they are linked/integrated together under the proposed framework umbrella. The framework is E2E (End to End) starting from the client gap assessments for the software application/project behavior/state against new style of IT services' best practices to asses current maturity level of the project/software application delivery which done via maturity calculator tool. Based on the outcomes from the maturity calculator gap assessment, set of actions are proposed along with CSFs (Critical Success Factors)/KPIs (Key Performance Indicators) to close current gap and facilitate move to the next level of maturity. These actions act as a transformation plan for the project in general and software applications/solution in specific towards the end state of new style of IT services. The framework afterwards provides push-button approach with fully automated setup toolkit to enable not only transforming the legacy/current project environment and infrastructure, but also migrating the current solution including software applications and incorporated data, towards the newly build cloud infrastructure with fully integrated set of tools based on the project needs/scope and overall project life cycle.

To ensure continual improvement and sustainability of the project and software delivery in seamless manner, delivery model with set of processes and models should be designed to enable smooth and seamless transition towards the new delivery approach and better communications between the tools and resources. Thus the focus in this section is to share and highlight the details of each and every component of the proposed framework:

- **MATURITY CALCULATOR**

The proposed calculator [25] for the DevOps maturity is one of the innovative and quantitative tools that aims to quantify the DevOps maturity for any organization or project with an organization in terms of set of capabilities/criteria. From which the tool uniqueness to shift the qualitative analysis [15] to measure the organization DevOps maturity into quantitative approach Through this quantitative approach, the organization/project DevOps maturity is assessed/measured against set of capabilities which selected carefully to cover the full DevOps landscape. The DevOps maturity consists of 14

capabilities that form the major components where any organization should adopt to follow DevOps delivery model. These capabilities vary between operational, delivery, governance, management, communication and process aspects. Each one of these capabilities assessed against different criteria to measure the different dimensions of the corresponding capability. DevOps maturity levels described in the previous section is utilized to measure the maturity level of each capability criteria. Each criteria has a standard description against each maturity levels (level 1 to level 5) where organization or project should meet to achieve such maturity level of the corresponding capability criteria.

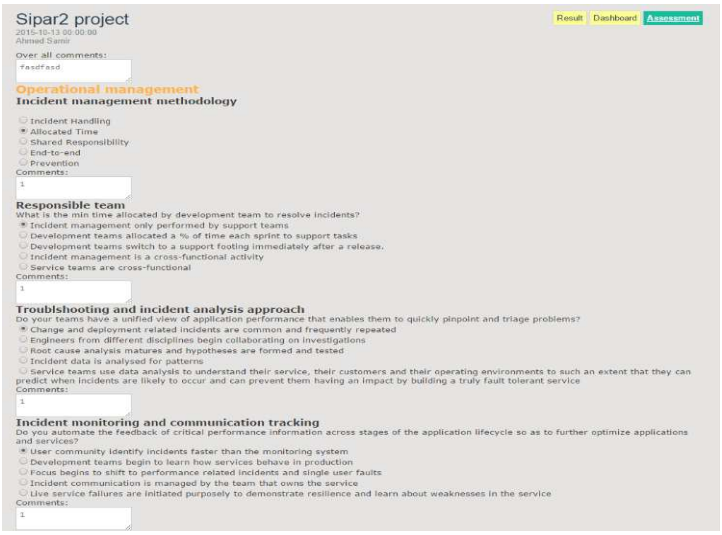

Figure 2: Maturity Calculator Capability and Criteria Interface

The maturity calculator is cloud-based application that captures the project/organization data/behavior against the 14 capabilities along with their corresponding criteria. Where User needs to select the best match between his/her project behavior and one of the listed DevOps criteria options, where each one is mapped by the tool to quantitative score according to the maturity level. Each criteria option is mapped against maturity level. Once user enters the project/organization behavior against all the 14 capabilities, the maturity calculator then calculates the overall score for the current project behavior mapped against DevOps standard represented with the 14 capabilities as mentioned above. Tool then maps the overall score which represents the project maturity level against thresholds set by the organization for each and every maturity level according to their business goals/objectives.

Maturity calculator produces different statistics to evaluate the project/organization maturity against the different capabilities. As shown in first pie chart figure 2, the project overall score is distributed to show the percentage capability against the different maturity levels. Based on our use case there is around 45% of the total capabilities under initial maturity level, 16% in the managed level, 12% in defined level, 4% measured level and 2% in the optimized level. This pie chart is vital for the organization to assess the current gaps and which areas that need more focus through the transformation plan towards the DevOps delivery plans.

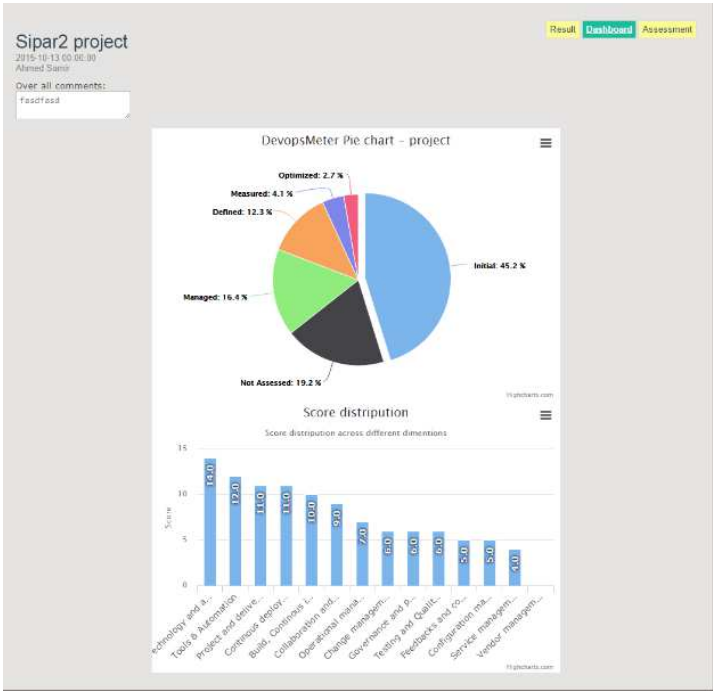

Figure 3: Maturity Calculator Statistics Results

The maturity calculator is cloud-based application that captures the project/organization data/behavior against the 14 capabilities along with their corresponding criteria. Where User needs to select the best match between his/her project behavior and one of the listed DevOps criteria options, where each one is mapped by the tool to quantitative score according to the maturity level. Each criteria option is mapped against maturity level. Once user enters the project/organization behavior against all the 14 capabilities, the maturity calculator then calculates the overall score for the current project behavior mapped against DevOps standard represented with the 14 capabilities as mentioned above. Tool then maps the overall score which represents the project maturity level against thresholds set by the organization for each and every maturity level according to their business goals/objectives.

Maturity calculator produces different statistics to evaluate the project/organization maturity against the different capabilities. As shown in first pie chart figure 2, the project overall score is distributed to show the percentage capability against the different maturity levels. Based on our use case there is around 45% of the total capabilities under initial maturity level, 16% in the managed level, 12% in defined level, 4% measured level and 2% in the optimized level. This pie chart is vital for the organization to assess the current gaps and which areas that need more focus through the transformation plan towards the DevOps delivery plans.

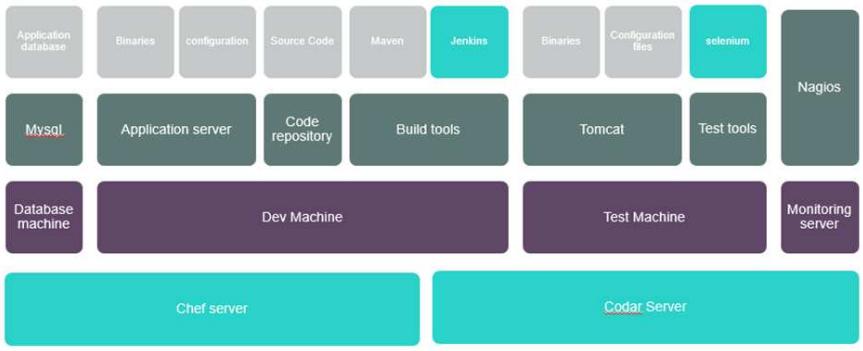

Figure 4: Environment Setup Toolkit Oveview

I). TOOLKIT PRE-REQUISITES

- In order to enable the setup toolkit, 'chef' and 'Codar' must be installed. This should be done once per the entire organization/account.
- 'Chef' and 'Codar' are integrated through creating a 'Chef' providers into the 'Codar' sever.
- 'Chef' cookbooks should be downloaded from 'Chef' marketplace and uploaded to 'Chef' server.
- Import 'Chef' cookbooks to 'Codar' as cookbooks to be able to provision machines from 'Codar'.
- Setup the network connectivity between 'Codar' and 'Chef'
- Handle the proper firewall configuration between the two servers of 'Codar' and 'Chef'
- Ensure proper connectivity between both servers of 'Codar' and 'Chef' and target machines on which other tools will be provisioned.

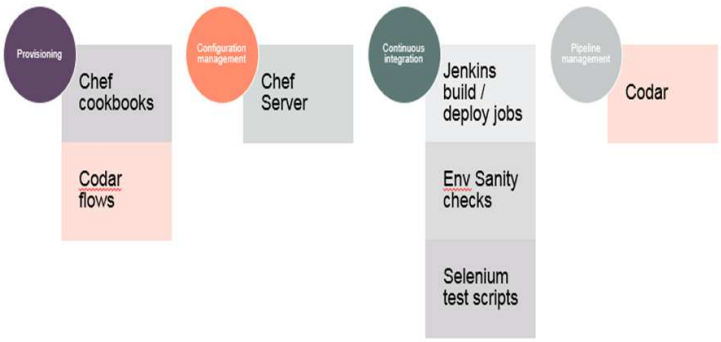

Figure 5: Environment Setup Toolkit Components

The approach is to automate the whole lifecycle of Application lifecycle management from servers' creation till deployment and testing the code change on production environment. The toolkit has for provisioning and environment creation 'Chef' cookbooks and 'Codar' workflows, so all servers' creation process will be fully automated. For build and deployment there is 'Jenkins' which build the code and then deploy it on the target environment. Once code is build, environment sanity checks scripts need to be triggered to check the environment status before running the automated test scenarios. Then afterwards 'Codar' will run for pipeline management.

II). OBJECT ORIENTED TOPOLOGY DESIGN

Once Chef Cookbooks are imported into Codar, provisioning environments becomes much more fun.

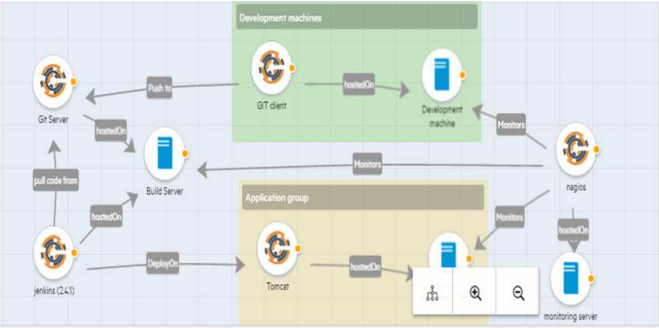

Figure 6: Topology Design Recommended for Proposed Framework

Through creating a new Object Oriented topology design and start creating environments out of it. After that, Topology designs can simply be exported and imported in any 'Codar' installation. But to use such topologies, we need to have a working chef provider configured. Following figure 5, the topology design recommended by the proposed framework. This design could be updated accordingly to match the business and technical needs of the project and/or organization.

III). CONTINUOUS INTEGRATION

To demonstrate how the setup toolkit is working for the proposed framework, I'll show three different use cases for couple of different tools/applications that will work automatically and under management of 'Codar'/'Chef' as described in the previous sections. The couple of tools are 'Git' which is configuration management tool [30], and 'Jenkins' which is automatic build tool for application source files [29].

Git Repository

- **Repository Initialization**

Through this phase, the objective is to initialize the 'Git' repository for tracking of all files in the 'Git' directory, and will refer to this directory as a working directory. Tracking is managed inside 'Git' directory which created by executing initialization command.

- **Recording Changes to the Repository**

Files can be under one of the following categories that will be managed by the 'Git' repository.

- Untracked → This file exist in the working directory, however we choose not to track it, that's to say if any changes happened to this file will not be detected by Git..
- Unmodified → This file exist in the working directory, but no changes were done on it.
- Modified → This file exist in the working directory, changes were done to it and need to be added to the staging area for the next commit.
- Staged → This file has been modified and staged in order to be saved as new version with the next commit.

Using different type of actions/command, software developer engineer will be able to:

- Check files status to assess if there is any modified or tracked files.
- Add a new file to our repository (working directory) and see how 'Git' will act based on this.
- Clone the repository or get an offline working copy within the same folder name as checked out from the main repository.
- Post commit configurations to trigger Jenkins build.

Jenkins Repository

Jenkins is used to build and test developed software projects continuously making it easier for software developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by providing powerful ways to define your build pipelines and integrating with a large number of testing and deployment technologies.

For Jenkins Installation and how to set up your environment, this will be covered in another Cook book, for now our scope will be creating a Jenkins job to be triggered whenever new code is committed to our repository.

For now we have two servers, CI server (where Jenkins is installed and will handle all build scenarios) and Application server (This will host the code repository and will be used to deploy the application after build with Jenkins)

Figure 7: Jenkins Repository Configuration

2.1. Set pre-authorized authentication between two servers. Application server will be identified to Jenkins server as node.

2.2. Add node name, this node will be considered slave node to Jenkins (CI) server.

2.3. Tune each property specs to the exact needs of software project, like number of executers and remote file path where Jenkins related properties will be used to connect to the target slave.

2.4. Launch slave agent you has just set, once you have your agent up and running then you can build jobs and dedicate it to this particular slave.

2.5 Create Jenkins job to be triggered automatically after code change for automatic build.

2.6. Automatically once any commit has been done at the 'Git' repository, after this job completes successfully it will run another Job that can test functionality of the deployed 'Ant' project [33] to automatically do a unit testing for the build code and be ready for automatic deployment on the target environment.

Delivery Model

Delivery model consists of set of use cases that model how software 'delivery and management' are constructed under the new style of IT strategies/vision through different aspects. The selected uses cases are selected based on the critical zones/areas through the software life cycle from environment setup towards operational delivery and troubleshooting as follows:

- Building New NSIT integrated environment (all layers including infra, DB, applications, Web/portal).
- Infra structure troubleshooting case (problem impacting the infra and all dependent layers from application and DB layers).
- Application deployment use case (consolidate all the interactions between teams/tools/processes throughout the deployment of any software change).

I). BUILDING NEW NSIT INTEGRATED ENVIRONMENT

This use case/model focus on modeling the sequence and workflow of setup any new integrated environment This includes the infra structure/virtual machines/cloud setup along with the Data Base (DB), software application/s and web/portal setups under the NSIT domain. Through next figures, I'll show and describe the different segments of this use case/workflow to show the interactions between different components of the model including tools, resources, and processes as per below figures (Figure 8, Figure 9, Figure 10, Figure 11). The first segment of this NSIT environment setup is focused on the Infrastructure setup of the whole integrated system based on how system will be hosted (ex. Virtual m/c or cloud based). Thus the use case starts when the cloud architect creates the cloud system servers from the cloud provider and pass then to operations team to validate and verify workability as per project/system specs. Once sanity checks validated, the automation engineer creates the Infra structure configuration management server to manage the different Infra structure Configuration Items (CIs) to ensure version control is concrete and controlled. Afterwards the automation engineer builds the boot strap servers using the infra CM created in the previous step of the use case to be utilized for automating the build of the other infra structure and environment setup.

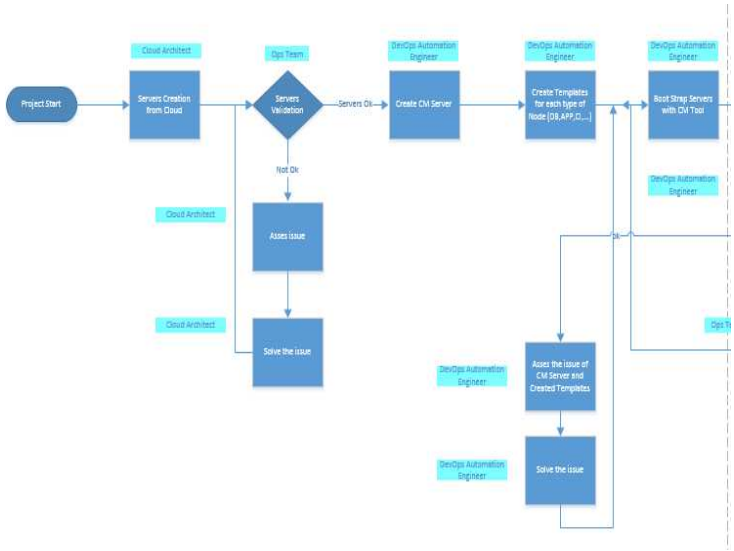

Figure 8: Setup New NSIT Use Case – Infra Segment

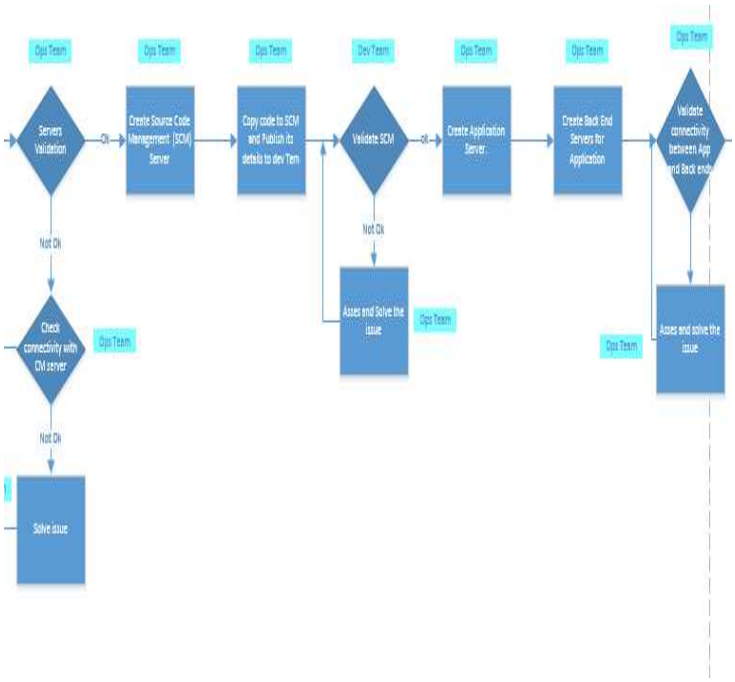

Figure 9: Setup New NSIT Use Case – Appliaction

The second segment of the NSIT setup use case starts to validate the created bootstrap servers build in the first segment and push any encountered m/c build issues to the automation engineer to fix. If no issues encountered, the created server delivered to the operations resources to build the source code (software application) configuration management (SCM) server then copy over the application code into the SCM. Once the source code is published on the SCM, development team starts to see/maintain/develop. The use case ensures to validate the build phase of the SCM before deliver to the next phase of the NSIT setup workflow through the operations team. Once SCM is build, the operations team starts to build the DB and application servers including both the front end servers (wep/portal) besides the backend servers (DB, other integrated systems, Load balancers). It will be delivered to the next segment till the connectivity between the front end application server and its backend are validated and checked.

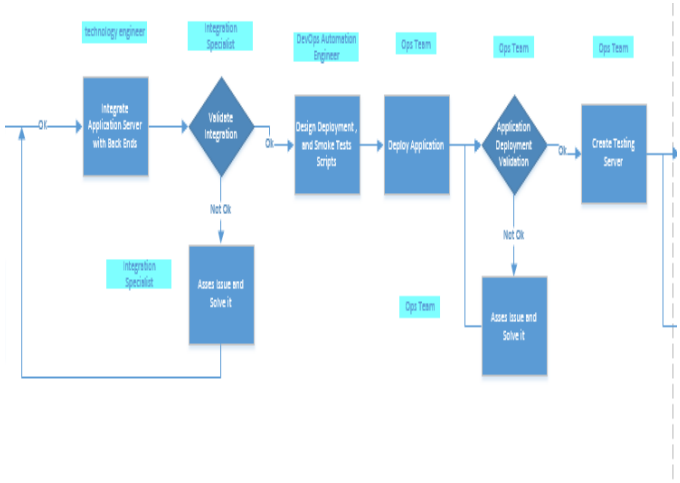

Figure 10: Setup New NSIT Use Case – Integration

The third segment of the NSIT setup use case starts to integrate the different pieces of the system together and once validated, the automation engineer designs and develop the deployment scripts and smoke test scripts to be able to deploy the application into the integrated environment. Once deployed application is validated by the operations team, it can be delivered to the next phase to build the testing server.

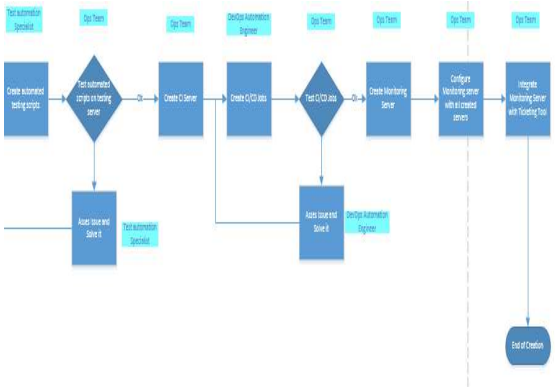

Figure 11: Setup New NSIT Use Case – Integration

The forth segment of the NSIT setup use case starts to validate the testing server after being built on the previous use case segment to start deploying the automation testing scripts. Once this is delivered the complete integrated environment is completed and ready for building the continuous integration (CI)/continuous Deployment (CD) servers which will be responsible for automating/promoting the different artifacts/source executables into the production environment once code is checked in by the development team. This ensures the checked in code will be automatically build and tested before being deployed into production. Last and final step in setup the integrated NSIT environment is to build the Monitoring and ticketing tools used by the operations team to monitor the application performance issues to ink it with the ticketing tool to fire incidents with proper severity/priority to operations team to fix or raise to revert back to development in case of code change is required to fix encountered issue.

II). INFRA STRUCTURE TROUBLESHOOTING

This use case/model focus on modeling the sequence and workflow of the Infrastructure troubleshooting and show the impact on the different processes, tools, resources, teams and communication to overcome the encountered issue in seamless and smooth manner as per below figures (Figure 12, Figure 13).

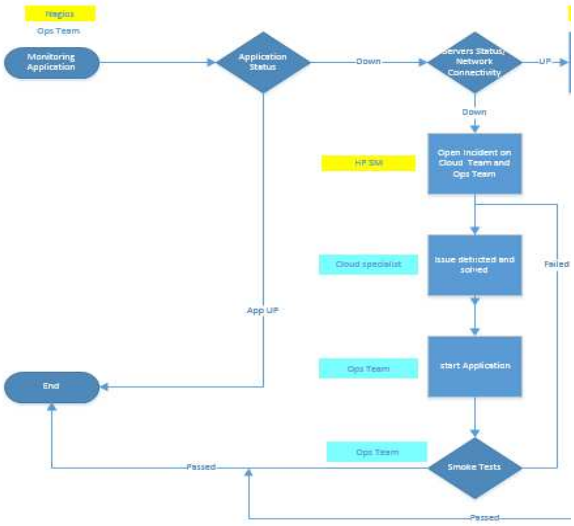

Figure 12: NSIT Infra Troubleshooting – Infra Checks

The first segment of this use case starts when the monitoring tools catch/detect an issue within the integrated system/environment, and once happened it fires a ticket via the Incident management ticketing system towards the respective team based on the issue classification and application detected. Once ticket is fired the operation team starts checking the network connectivity, in case application is down. If issue related to network, ticket will be forwarded to infrastructure/network team to fix the issue and then sync with operations team to startup the application. Once smoke tests/sanity checks are passed the ticket will be closed.

The second segment of the troubleshooting use case starts by checking the application/backend systems in case issue remains after segment one is covered. This starts when the ticket is assigned by the ticketing system on the operations team to check the application and back end systems status via the sanity checks/automating scripts that detect/fix the issue. Once issue detected/fixed either in the application or the backend systems, sanity checks are triggered to validate the resolution approach before updating/closing the ticket and delivered.

In case issue is not within wither the application or the backend systems then the operations team forwards the ticket to the development team to check the application source code through debugging tool to detect/fix the issue. Once development team detect/fix the problem, they need to check in the new code via SCM and promote to production after proper testing/QA cycles passed. Once passed the ticket can be closed and use case will then be covered.

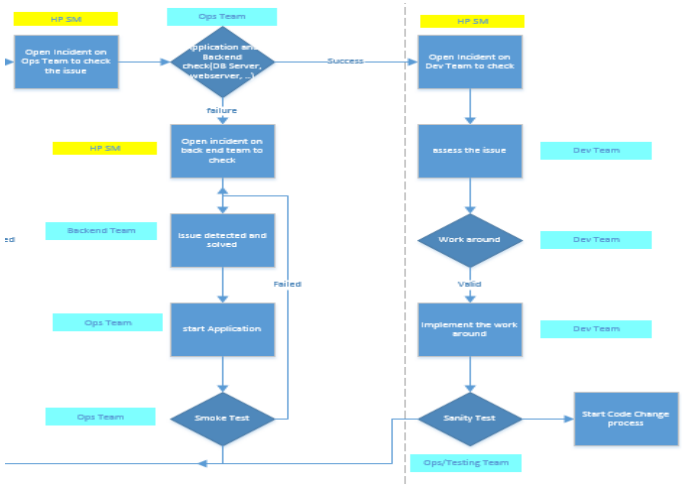

Figure 13: NSIT Infra Troubleshooting – Application Checks

III). APPLICATION DEPLOYMENT/CODE CHANGE

This use case/model focus on modeling the sequence and workflow of the application deployment on the different environments using specific set of tools highlighted in amber as per below figures (Figure 14, Figure 15).

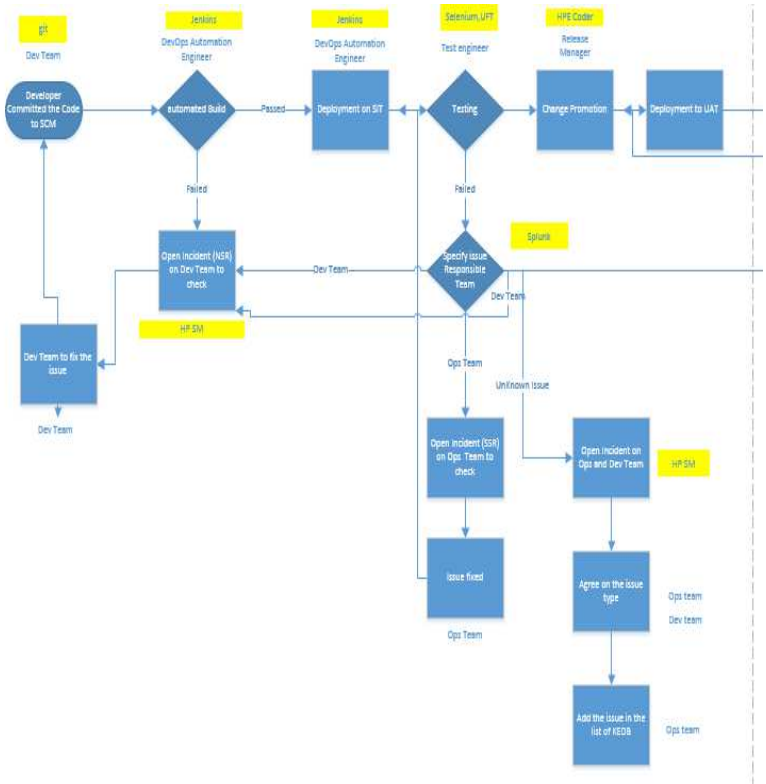

Figure 14: NSIT Software Deployment – Code Change

The first segment of the deployment use case starts when the development team check in the code via SCM using GIT tool. Once code is committed, the automated build will be triggered using 'Jenkins' tool. Tool till fire a ticket on the development team for any issue with the build/build failed. Once build of the newly added software code is passed, the application will be deployed to SIT (System Integration Test) environment using 'Jenkins' to complete the SIT tests.

Automated testing scripts triggers the regression testing via 'Selenium' tool. In case of any issue with the regression testing, the tool fires a ticket on the responsible team, either development to fix the code responsible for the failed test case and repeat the cycle by committing a new code, or raise a ticket on the operations team to fix the issue by changing the application configuration and/or infra design parameters. Once all test cases passed on the SIT, the 'Codar' tool promotes the code into the UAT (User Acceptance Test) environment.

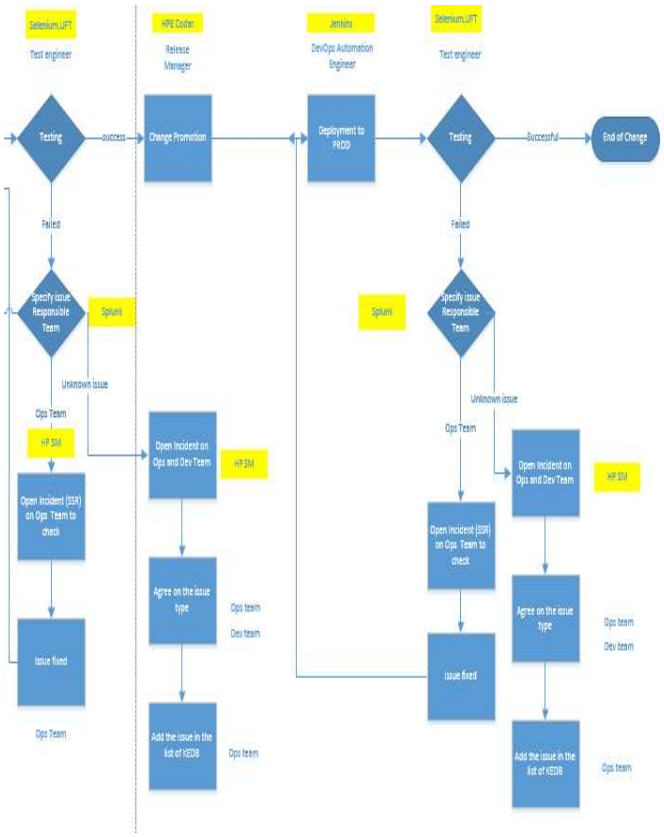

Figure 15: NSIT Software Deployment – Deployment

The second segment of the deployment use case continue once code is deployed to UAT environment where same kind of testing is done on production like environment to simulate performance and application behavior on the production environment. In case of any issue detected the 'Splunk' tool will be able to detect and forward to the respective team either development team to take the proper corrective action to fix the code responsible for the issue and repeat the deployment cycle from the beginning, or the operations team to fix the issue vi the SM 'Service Management' tool. Once UAT test is passed the 'Jenkins' tool deploys the code automatically into production where final set of production test cases run to ensure no issues will be faced by the end users on live environment. In case of any issues the 'Splunk' tool [35] forwards to the corresponding team.

4. FRAMEWORK VALIDATION

The proposed NSIT framework PoC (Proof of Concept) is built to validate and show the improvement gained/achieved from adopting the NSIT best practices, tools, and communication toolkit and resources skill sets/caliber. This PoC is implemented on real project/application data which was initially hosted on legacy desktop servers. To show the ROI/Value behind the proposed NSIT framework, I used #hours/effort saved by using the proposed framework versus the

manual implementation of the same type of work. With the new approach of NSIT as in figure 16.

The specification of the testbed or system under test is based on the following assumptions as:

- Four servers (Dev, Test, Nagios, Database)
- One target is done over a java application.
- Average no of code changes is 4 changes per week
- Average no of environmental changes is 5 changes per week
- 3 ESXI servers with 6 virtual machines with Centos OS
- Installed chef server for provisioning of all servers with development of cookbooks for each machine creation to make the creation automated
- Codar topologies

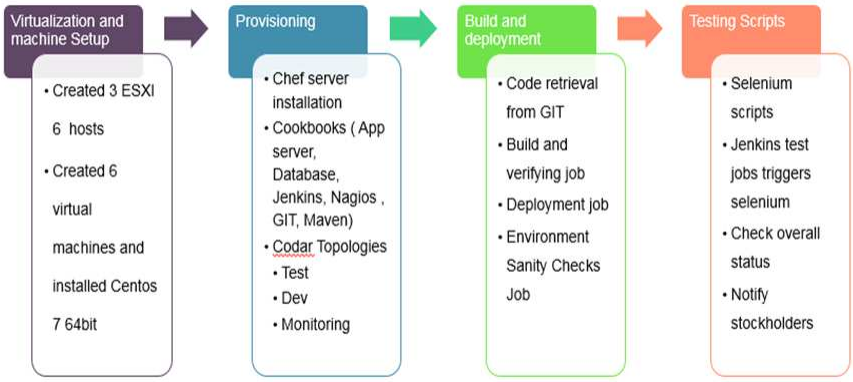

Figure 16: NSIT Proof of Concept of the Proposd Framework

Implementation processes to setup the NSIT toolkit are as follows:

- Build set of virtual m/c to host the new integrated system/applications under the NSIT framework (6 m/c of the following technical configurations).
- 6 Virtual m/c is basically to simulate three different environments (Dev, SIT, Prod).
- Configure and setup the different NSIT tools (CM, SCM, Jenkins, Git, Codar, SM, Chef).
- Build the application server (front end and backend servers including the DB, LBs (Load Balancers).
- Integrate the application servers (frontend and DBs) hosted on the virtual m/cs.
- Deploy and configure the DB on the new build DB server.
- Deploy the application on the new build application server and ensure the connectivity between the systems is as expected.
- Build the CI/CD (Continuous Integration/Continuous Deployment) servers and link with ‘Jenkins’ and ensure they

are linked with the integrated system of application and DB.

- Build the testing server and deploy set of test cases using 'Selinum' [34].

Framework validation objective is basically to check the automatic full software life cycle management starting from committing a new code into development environment till being delivered/deployed into production/live environment using the configured and tools setup on the NSIT environment. This starts when the software development engineer check in/commit the code into the development environment using 'Git' tool. 'Jenkins' triggers the automatic build afterwards to pick the source code changes into the target package and then deploy/promote into the SIT environment. Once being deployed, 'Selenium' automatically trigger the regression testing. Two paths validated in this case, the FAIL path when test case is failed, where 'Selenium' fire ticket via 'SM' against the respective development/operations team as configured in the 'SM' profile done through the configuration setup as mentioned above. The other path which validated is the PASS path where all test cases are succeeded when 'Codar' then pick and promote the package into the life/production environment automatically.

This analysis can be shown with the following comparative analysis between the manual approach and automated/NSIT approach:

I). ENVIRONMENT SETUP (ONE MACHINE)

Table 1: Enviroment setup effort estimation

Activity	Manual(H)	Automated – NSIT (h)
OS configuration	2	5
Application installation	2	2
Application configuration	2	2
Machine bootstrap	-	15 min * 4 = 1
Total	**6 X 4 =24 Hours**	**10 ours**

II). ENVIRONMENTAL CHANGES/CONFIGURATION

Table 2: Enviroment change effort estimation

Activity	Manual(min)	Automated – NSIT (min)
Environmental change	15 per change	15
Total	**15 X 4 X 5 = 300/Week**	**15 5 = 75/Week**

III). CODE CHANGE/DEPLOYMENT

Table 3: Code Change Effort Estimation

Activity	Manual(min)	Automated (min)
Code build	15	10
Code deployment	30	15
Sanity checks	30	5
Test scenarios	120	15
Communication overhead	60	0
Total	**17 Hours Per Week**	**3 Hours Per Week**

To show the value and ROI (Return on Investment) for the proposed solution we can utilize the data listed in tables 1, 2 and 3 via following statistical figures:

The statistical figures (Table 1, Table 1, and Table 1) sumerize the value from using the proposed NSIT framework from differrent dimentiones starting from setup the new environment, Environment configuration and code change/deployment for both the manual and automated/NSIT approaches. As shown in Figure 17, the ROI for the automated/NSIT approach save almost 70% of the total time compared with the manual approach. Figure 18 represent the ROI for the environment configuration which clarify that NSIT approach saved almost 75% compared with the manual approach. And last but not least comparing the ROI for code change as shown in Figure 19 between the manual/automatic show saving of 80% compared with the manual approach

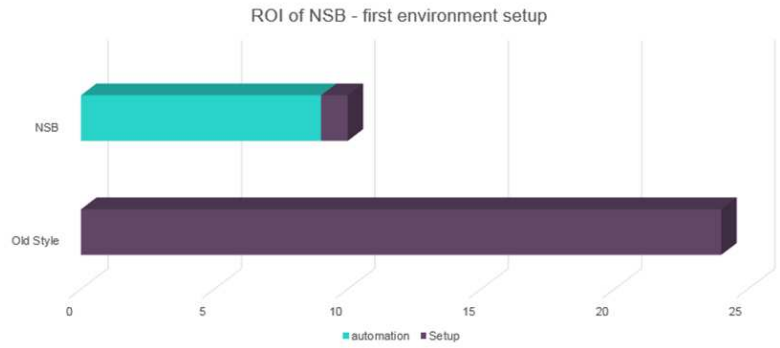

Figure 17: ROI for First Environment Setup

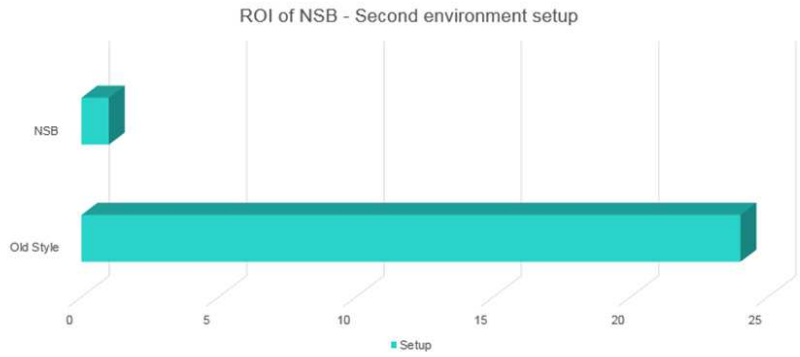

Figure 18: ROI for after First Environment Setup

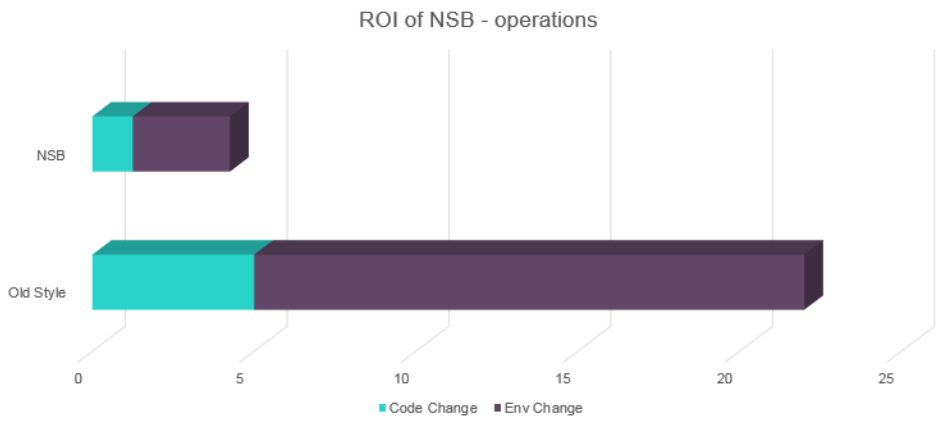

Figure 19: Roi for Operations for Single Deployment

5. CONCLUSIONS

Work presented in this paper is part of series of articles presented under the NSIT (New Style of IT) domain/framework [25-28]. The article highlights the NSIT framework and how it can fit within the work for application modernization [15]. The proposed framework starts by assessing the current application behavior and life cycle against the NSIT best practices using he maturity calculator. Outcomes of this gap assessment drafts the next steps for either transformation recommendations or complete environment setup for the applications and the integrated system into the NSIT environment. The proposed push button approach for automated environment setup toolkit clarify how to fully automate the environment setup with full set of tools that serve project/application life cycle. Then the framework provides set of use cases to guide the delivery model throughout main basic life cycle scenarios like environment setup, application deployment, and infrastructure troubleshooting. The models/use cases presented in this article show the different dimensions and aspects of the software life cycle management on the NSIT environment. These framework components show how the NSIT framework helps currently IT organizations to first satisfy their client demand/needs and face the current challenges within the competing market. The innovative framework articulates the different resources available from business/technical processes, tools, resources, skill sets, and caliber of the involved teams, communications to deliver in seamless and smooth manner towards the final delivery goal. This raised especially for satisfying the high traffic/demand from the clients especially with virtualization, cloud computing, mobility, DevOps, social and big data start to populate and boom. Future work is to extend this NSIT framework to utilize the big data analytics resources to facilitate better automation, reporting and data/trend analysis of the system data.

REFERENCES

1. D. DeGrandis. Devops: So you say you want a revolution? Cutter IT Journal, 24(8): 34{39, 2011.
2. D. Feitelson, E. Frachtenberg, and K. Beck. Development and deployment at facebook. IEEE Internet Computing, 17(4):8{17, 2013.
3. L. Fitzpatrick and M. Dillon. The business case for devops: A five-year retrospective. Cutter IT Journal, 24(8):19{27, 2011.
4. S. Hosono and Y. Shimomura. Application lifecycle kit for mass customization on PaaS platforms. In Proceedings - 2012 IEEE 8th World Congress on Services, SERVICES 2012, pages 397{398, Honolulu, HI, 2012.

5. J. Humble and J. Molesky. Why enterprises must adopt devops to enable continuous delivery. Cutter IT Journal, 24(8):6{12, 2011.
6. B. Key worth. Where is it operations within devops? Cutter IT Journal, 24(12):12{17, 2011.
7. B. Kitchen ham. Procedures for performing systematic reviews, 2004.
8. O. Akerele, M. Ramachandran, and M. Dixon. System dynamics modeling of agile continuous delivery process. In Proceedings - AGILE 2013, pages 60{63, 2013.
9. S. Bang, S. Chung, Y. Choh, and M. Dupuis. A grounded theory analysis of modern web applications: Knowledge, skills, and abilities for devops. In RIIT 2013 - Proceedings of the 2nd Annual Conference on Research in Information Technology, pages 61{62, 2013.
10. L. Bass, R. Je_ery, H. Wada, I. Weber, and L. Zhu. Eliciting operations requirements for applications. In 2013 1st International Workshop on Release Engineering, RELENG 2013 - Proceedings, pages 5{8, San Francisco, CA, 2013.
11. D. Cukier. Devops patterns to scale web applications using cloud services. In Proceedings - SPLASH '13, pages 143{152, Indianapolis, Indiana, USA, 2013.
12. P. Debois. Opening statement. Cutter IT Journal, 24(8):3{5, 2011.
13. A. Schaefer, M. Reichenbach, and D. Fey. Continuous integration and automation for devops. Lecture Notes in Electrical Engineering, 170 LNEE: 345{358, 2013.
14. W. Shang. Bridging the divide between software developers and operators using logs. In Proceedings - International Conference on Software Engineering, pages 1583{1586, 2012.
15. S. Stuckenberg, E. Fielt, and T. Loser. The impact of software-as-a-service on business models of leading software vendors: Experiences from three exploratory case studies. In PACIS 2011 - 15th Paci_c Asia Conference on Information Systems: Quality Research in Pacic, 2011.
16. B. Tessem and J. Iden. Cooperation between developers and operations in software engineering projects. In Proceedings - International Conference on Software Engineering, pages 105{108, 2008.
17. M. Walls. Building a DevOps Culture. O'Reilly Media, Sebastopol, CA, 2013.
18. J. Webster and R. T. Watson. Analyzing the past to prepare for the future: Writing a literature review. MIS Q., 26(2):xiii{xxiii, June 2002.
19. A. Le-Quoc. Metrics-driven devops. Cutter IT Journal, 24(12):24{29, 2011.
20. M. Loukides. What is DevOps? O'Reilly Media, Sebastopol, CA, 2012.
21. S. Neely and S. Stolt. Continuous delivery? easy! Just change everything (well, maybe it is not that easy). In Proceedings - AGILE 2013, pages 121{128, 2013.
22. B. Phifer. Next-generation process integration: CMMI and ITIL do devops. Cutter IT Journal, 24(8):28{33, 2011.

23. H. Pruijt. Multiple personalities: the case of business process reengineering. Journal of Organizational Change Management, 11(3):260{268, Jan. 1998.

24. J. Roche. Adopting devops practices in quality assurance. Communications of the ACM, 56(11):38{43, 2013.

25. S. Mohamed: DevOps Maturity Calculator DOMC - Value oriented approach, International Journal of Engineering Science and Research, Vol 2, Issue 2, PP 25-35.

26. S. Mohamed: DevOps shifting software engineering strategy-value based perspective, International Journal of Computer Engineering, Vol 17, Issue 2, and PP 51-57.

27. S. Mohamed: GOAL ORIENTED DEVOPS TRANSFORMATION FRAMEWORK – METRIC PHASED APPROACH, International Journal of Current Research Vol 8, Issue 3, PP 28307-28313.

28. S. Mohamed: New style of software lifecycle strategies – Use Case perspective, International Journal of Management, Information Technology and Engineering, Vol 4, Issue 3, and PP 99-114.

29. https://wiki.jenkins-ci.org, Apr 2016.

30. http://www.tutorialspoint.com/git, Mar 2016.

31. https://www.youtube.com/watch?v=fVUoWqmuYJM, Mar 2016.

32. https://docs.chef.io/install_server.html, Feb 2016.

33. http://www.tutorialspoint.com/ant/, Jan 2016.

34. http://www.tutorialspoint.com/selenium/, Feb 2016

35. http://www.splunk.com/view/SP-CAAAHSM, Mar 2016.