AUTOMATION OF DEVOPS IN EMBEDDED SYSTEM DEVELOPMENT: TOOLS, CHALLENGES AND SOLUTION

[1]Ravikumar N Pareet
[1]Student
[1]Computer Science and Engineering
[1]R V College of Engineering, Bangalore, India

Abstract: - DevOps is a very important culture in the software industry. It is a set of practices in the software engineering domain. It automates the process between software development and operation teams, in order to build, test and release software faster and more reliable. DevOps enables the better communication between development and operation team which is responsible for creating and maintaining the infrastructure. It includes agile planning, Continuous Integration (CI), Continuous Delivery (CD) and monitoring of the applications. This paper aims at systematically reviewing the state of art, of the DevOps tool-chain in embedded software development and identifies the challenges and practices. Various approaches and tools are used in order to develop embedded software faster, as quality and reliability are very critical factors in embedded system. By considering the hardware dependencies and interfacing constraints some of the approaches defined are 1) Automating the continuous integration to reduce build and test time 2) Static code analysis to analyse the structure of the code and detect anomalies which leads to real bug 3) Automation of continuous deployment 4) Automate the deployment process to drastically reduce the time and enable faster feedback.

***IndexTerms* - DevOps, CI/CD, Jenkins, Git, TICS, klocwork, TFSA, Docker**

I. INTRODUCTION

DevOps is a compounded term which can be expanded as development and operations. DevOps is another new wonder in the world of programming, automation, virtualization, stressing coordinated effort and new instruments that extends advancements in operations and the development opportunities [1]. It focuses on improving the communication, collaboration among teams and integrating between the operations and the development of teams. It is an efficient way to improve the satisfaction of the customer and to boost the software systems quality. Increasing the satisfaction of the customer is the main goal as it forms the base of any software that is being distributed across globally. This could be achieved by obtaining faster delivery of the product with the requested features. DevOps has the capability of offering continuous improvement and continuous development so it must offer continuous satisfaction for its customers. It helps the software industries in both non-technical as well as in technical way in order to improve the receptivity of the needs of the customer by releasing the software's in periodic cycles and automating the software releases.

Risks can be reduced by frequent releases that are related to deployment and results can be evaluated in a faster way with respect to the transformations in the software, configurations. Quicker feedback enables newer data to be added that will enhance the process of software development [2]. Deployment of new software features into the existing system has increased from multiple releases in a month to multiple releases a day increasing the competition among the industries particularly in the field of applications where there is great demand delivered using SaaS that is Software as a Service that are streaming over the internet. It tries to bridge the gap between operations and development teams [3]. DevOps is not yet adopted in the field of embedded systems. It is very easily adaptable in the domain of web as the concept of virtualization will greatly help the developers to extract the infrastructure of the software. Configuration tools supports the development, followed by testing that will be reflected in the production. Software for configuration management is used with automatic verification and automatic validation software. This will boost the confidence levels for the first-time deployment of the newer features for the software that are being deployed in the production. Software's are constantly being deployed every day in the field of IT. It mainly depends on the reliability and the stability of the software.

II.EVOLUTION OF DEVOPS:

Compilation of the code is a vital advance of software industry in which the primary source code is being changed over to executables [5]. In Continuous Integration, and Test Automation (TA), designing patterns, refactoring of the code and other different methods are being utilized by huge endeavors for upgraded quality and venture productivity [6]. Reddy and Rauf in [7] have spoken about the significance of Test Automation in the software IT industry and proposed a calculation for planning experiments and Test Automation. Patrick Day in [8] has spoken about the design of the Test Automation for the agile model. In view of interest, planned or based on triggering of the event automation can be accomplished. Bhanu in [9] has worked on a Software model for testing in the cloud condition which forms the base for Continuous Integration. They additionally examined the testcases that were automated to run in an efficient and an easy way and in a productive less demanding way. They utilized Jenkins for making, planning jobs for the software. Dyanlin [10] examined the significance of controlling the version and managing the patch for the systems that were having automated frameworks and their usage utilizing viable innovations like ParamRev for following the changes in the parameter and ConfigRev for following the changes in the configuration. In [11] Preeti has examined the entire history, establishment and design of Jenkins. They likewise looked at other integration tools introduced in the market with Jenkins. In [12] Yanmei combined Maven, SVN Subversion with Jenkins and Android JUnit and suggested a way to deal with the scheduling of different unit tests as a single build. They suggested this way for android improvement ventures. Stephan in [13] has spoken about the automating the NFC-based IC tests by combining it with the Jenkins and NUnit.

III. DEVOPS TOOL CHAIN

Figure 1 shows the tool chain of DevOps. Tool chain is a set of tools which are used to accomplish continuous processes. The continuous processes may be continuous development, continuous testing, continuous integration, continuous deployment and continuous delivery. [14]

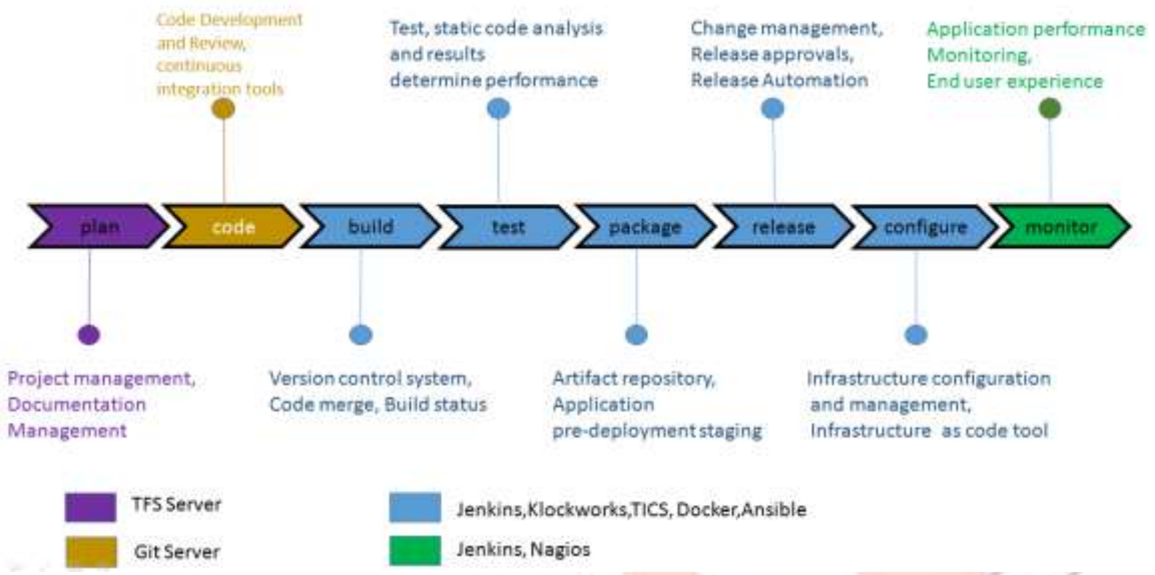
Code Development and Review, continuous integration tools
Test, static code analysis and results determine performance
Change management, Release approvals, Release Automation
Application performance Monitoring, End user experience
plan
code
build
test
package
release
configure
monitor
Project management, Documentation Management
Version control system, Code merge, Build status
Artifact repository, Application pre-deployment staging
Infrastructure configuration and management, Infrastructure as code tool
TFS Server
Git Server
Jenkins,Klockworks,TICS, Docker,Ansible
Jenkins, Nagios

Figure 1 DevOps Tool Chain

In this tool chain, TFS server is being used in the planning phase to maintain the documentation and the backlogs of the project. For development and reviewing of the code Git server is being used. Jenkins is used for continuous integration and continuous deployment. Klockwork and TICS for static code analysis. Ansible for configuration management and Docker for build and containerization. And lastly monitoring is done by Nagios.

A. Git Version Control:

Git is the most widely used revision or version control and source code management system. Version control system is a software that allows the software developers to work simultaneously and maintain the complete history of the developers work. Git is released under General Public License (GPL) open source decentralised version control system. Git overcomes the drawbacks of centralised version control system and has many advanced features in it than the previous versions. Git has a very good failover and backup mechanism. In case of failure of server, the restoration can be done by copying the data from any of the client's repository back to the server.

Some of the advantages of git are listed here.

- **Free and open source**: It is released under GPL open source license. Git tool can be used to manage the project without paying even a penny as it is an open source tool.
- **Fast and Small**: As it is decentralised version control system, most of the operations are done on the local system. This gives huge benefit in terms of speed. Software developers explicitly need not to be available for every operation that is being performed. Generally the size of the data that is present on the client side will be small.
- **Implicit backup**: The likelihood of losing information is exceptionally surprising as there are many duplicates of the data. Information exhibits on any customer reflects the same in the repository.
- **Security**: Git uses secure hash function, which most popular cryptographic hash function that is used to name and identify the object present within its database. Each document and commit will be registered and recovered by its checksum at each checkout. It is difficult to change the date, record, commit message and some other information from the Git database without the proper understanding of Git.
- **No Need of powerful hardware**: As git is decentralised version control system all heavy lifting happens on client side and not on the server side. Developers will not interact with server every time unless they need to push their code into the repository.

- **Easy Branching**: It uses lazy copies and branch mechanism. It works similar to that of a hard link concept in the UNIX file system. Branch mechanism with git is very simple.

B. Jenkins:

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software. It is an automation tool for continuous integration where developers periodically merge their code. It is a free source that can deal with any sort of constructs. Jenkins can be integrated with many a number of testing and deployment technologies. CI is an improvement practise that expects designers to combine code into a common database at generic intervals. Figure 1 shows a simple workflow of how the Jenkins works:

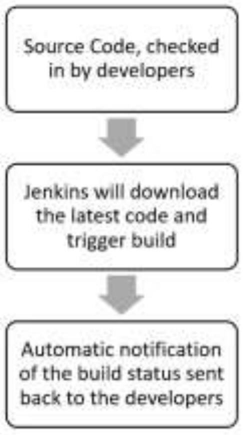
Source Code, checked in by developers
Jenkins will download the latest code and trigger build
Automatic notification of the build status sent back to the developers

Figure 2 Jenkins Workflow

C. Static code analysis:

The static code analysis is the most important testing approach in embedded software development that can be used to inspect all code paths and data flows across a program that has to execute without even actually running the program. Static code analysis is cardinal step in embedded system software development because even a small bug may cause huge threat to human life. The Software reliability is a very important factor. A small change in software may increase the overall risk of the system's reliability. As system grows in its functionality, especially in the embedded system, which is mostly controlled through software. This when combined with the hardware reliability has helped in migrating the system failure from hardware to software level.

Static analysis is an approach which helps to improve the quality and reliability of embedded systems software. Adopting the static-analysis tools and methods into the development process, it can provide significant reductions of failures in development, testing and customer platform. It is process of analysing the source code by retaining some of the predefined standards and avoids the overhead of writing and running test cases or incrementing the code. It also helps in automating the manual verification tasks.

There are various forms of static code analysis techniques of which some are built into compilers and IDE's itself and the rest are provided in most popular tools like:

- Lint
- TIOBE TICS framework
- Klockwork

Static code analyser can be used as plug-in to their IDE which helps the developer for identifying the errors in coding standards and semantics. It can also be integrated in the build environment.

TIOBE TICS tool and Klockwork together offers the industry's most complete automated static analysis, making software development much faster and reliable.

Klocwork:

Klocwork is one of the most widely used static code analysis tool to identify security, safety and reliability issues in C, C++, Java and C#. TICS framework support clockwork plug-in for abstract interpretation. Klocwork includes numerous desktop plug-ins for developers. It provides a dashboard for displaying the metrics and also for reporting. It enables the faster delivery of secure and reliable code.

Klockwork has the following feature:

- **Static Code Analysis** – It identifies the issues at the earliest.

- **Smart Rank** - It prioritizes and works on the issues that are important
- **Continuous Integration** – It can be used with auto build for increased scalability and performance
- **Application Security** – It helps in preventing malicious attacks
- **Code Refactoring**–It cleans up the code structure and reduces future costs to the development
- **Reporting and Metrics** - Understand and prioritize issues
- **Code Architecture** - Visualize and optimize software design

TIOBE TICS:

The TICS Framework integrates the quality data from different analysis tools into a stable environment. The data is stored in TICS quality database and are displayed at the department level via a web application called the TICS viewer i.e., TICS enterprise dashboard.

D. Docker:

Docker is a very popular container management service. It is a computer program which yields operating system level virtualization known as containerization. Docker uses resource separation properties of the Linux kernel such as kernel namespaces, union-capable file system and cgroups to allow isolated "Container" to run within a single Linux instance by avoiding the overhead of starting and maintaining VM's. Docker container doesn't require separate operating system to run like the virtual machine. It is capable of running on Linux kernel provided with all the facilities.

Docker is an open source platform used to automate the development cycle of the application. Docker confines the application program, run time and it's dependencies in a virtual container that could run on any Linux server. This enables flexibility and portability of the image which can be run anywhere. The significance of Docker is **Build**, **Ship** and **Run** anywhere.

Containerization technology can be used to speed up the development cycle in an embedded system. This can be achieved with some constraints by developing codes for Linux build which can be run on Linux operating system. The same code will be used for real hardware by building wrapper. It could make device specific calls or API's.

Some the key benefits of using Docker in Embedded software domain are:

1. It allows using the tool in local environment without the need to install them
2. It allows code to be checked in different forms of tool chains not worrying much about tools co-existence
3. It ensures the same tools would be used on build server as well
4. It allows to create virtual TCP/IP connections for different applications on the same machine
5. It allows installation of docker without infecting the development machine

Some Challenges:

1. Container technology can't be used on the embedded devices having memory constraints
2. Tested Linux build may not work on real hardware as it lacks many device specific APIs
3. It cannot be used on any embedded devices which do not support containerization
4. It can't be used on bare metal

IV. METHODOLOGY

The proposed methodology requires some software's that needs to be installed. The first is the operating system. Here Ubuntu 16.04 LTS server operating system is being used as this project needs two types of build one the Linux build and other is the RTXC build (a Real Time Operating System). Other tools that need to be installed are Jenkins, framework for automated unit tests, integration tests, static code analysis, supporting libraries and runtime environment, Jenkins plug-ins like Email-extension to get email notification and zipped logs and Docker to create the build.

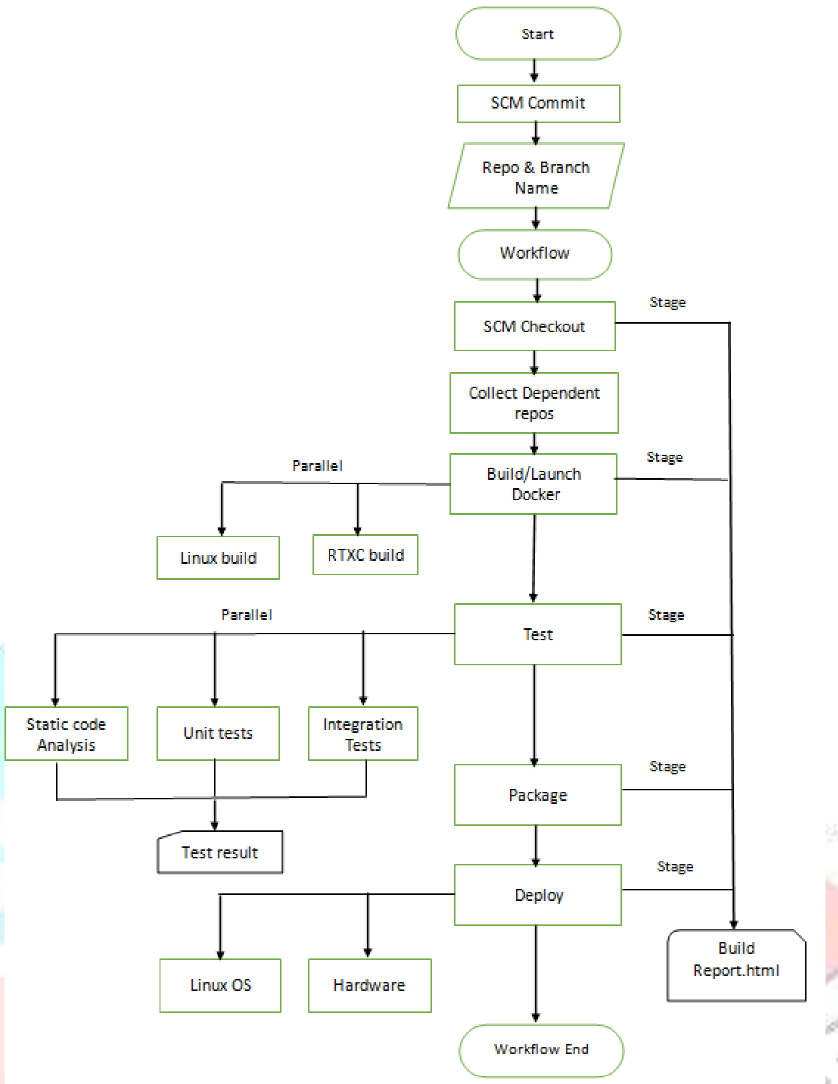

Figure 3 Workflow

Once the installation is complete, Jenkins is allowed to create a task for automated CI/CD which consists of sequence of stages and each stage may have many parallel executions. The Jenkins job is created with the help of the Git hooks. For every SCM commit, Jenkins job will be automatically triggered, this is because of the change in code in the repository. The SCM commit will be done on specific branch (Develop/feature branch). From here the actual workflow starts. SCM checkout is done in order to download the latest code from the repository. The next step is to build the code on the specific platforms like Linux or RTXC. Linux and RTXC build will be performed in a parallel. The next step is to do different testing on the build. Static code analysis, unit testing and integration testing are generally done on every build in parallel and the test results will be logged in an xml file. After successful testing, build is being packaged and will be deployed either on Linux OS ore on hardware (production environment). The results of each stage will be logged in BuildReport.html and also will be notified through email.

V. BENEFITS OF ADOPTING DEVOPS:

There are several benefits by adopting the DevOps in embedded domain where agile development is in practice. Embedded software is a program that is directly used or embedded into the hardware device. It is mainly written for particular hardware because of the computing capability of device that usually has processing and memory constraints to run. As embedded system domain software implementation is more depends on hardware or devices. The speed of the embedded software development may not be as fast as a web application domain. It has less delivery time so in this era of digitization and rapid change in technologies it is very much necessary for improving the business and also to satisfy the customers.[15]
Some of the important benefits include:

1. DevOps and Automation enables faster release and deployment cycle where agile development is in practice.
2. It Improves the communication between application development, business stakeholders and operations team.
3. It minimizes the implementation time of new features or services from time duration of months to minutes.
4. It improves the productivity of business and Operation teams
5. It is standardizing the process for easy replication and increases the speed of the delivery
6. It improves the quality, reliability and reusability of all system components
7. It improves the rate of success for digitization strategies and transformation projects

8. It satisfies customers and employees.
9. Minimize the deployment related to downtime

VI. CHALLENGES FACED WHILE ADOPTING DEVOPS

Another objective of this paper is to identify the challenges that will be faced by adopting the DevOps in embedded system software development. Some of the challenges that one may face when DevOps and automated CI/CD are integrated into an embedded domain includes:[16][17]

1. There may be hardware and interfacing dependency as they do not follow a modular approach and all or some of the components may depend on one another. So, the flow between the components needs to be kept independent of application and the architecture which may not be possible always.
2. The software used should be compatible with the hardware version that is being used
3. Integrating new tool with existing tools in the organization
4. Lack of proper automation tool in embedded systems domain
5. Software that requires cross compilation to the embedded platform
6. Limited visibility of production environment in embedded system domain
7. Lack of proper resources to configure environments
8. Culture and mind set of log timer in the organizations
9. Scalability and hardware resource constraints
10. Some legal and regulatory constraints

There are several challenges by adopting the DevOps of which some can be solved by using new open source tools like Ansible for configuration management, automation of deployment and monitoring.

Most of the embedded software is written in C/C++. Cross compilation should be done, multiple tool chains must be made used off to run binaries into targeted hardware or environment. The build infrastructure is generated and controlled by C/C++ Development Tooling project (CDT) in eclipse automatically, which is consisting of a set of Makefile that is generated and inserted into software projects itself.

There are no tools to deploy application onto the target environment. The required scripts needs to be loaded on a target platform over the network which had been written and used at some point of time, but were never maintained. Other way of doing this is manually uploading the code onto the board using USB stick. This increased the feedback cycles in software development. The entirely manual testing process is carried out over the period of week and multiple engineers were required to work on it and builds were performed infrequently.

VII. SOLUTION TO OVERCOME THE CHALLENGES FACED BY ADOPTING DEVOPS:

The agile methods have managed to improve productivity, reliability and delivery of the product in the software development industries. Automation is employed in the organization where agile is in practice, mainly to reduce human mistakes and to get faster feedback, to satisfy customers by delivering quality product at the earliest. Continuous delivery is set of process designed to ensure that code can be rapidly and safely deployed to production by delivering every change to the production environment. With automation and agile practices in the organization there exist some bottlenecks .The rootcause of this problem is, operation teams do not follow practices similar to that of the development team .DevOps breaks the bridge between development and operation teams so that huge time taken to collaborate between operation teams can be avoided or reduced completely. DevOps is a culture improves communication, collaboration and integration between software development and operation teams in the organization.[16]

Some of the solutions proposed for overcoming the challenges are

1. Initially find out all the hardware and software dependencies in the system and then prioritise the tasks.
2. Gain thorough understanding of the hardware and software architecture and ensure that the architecture follows a modular approach.
3. Collaborating developer team, operation team and testing team to reduce the communication gap between them.
4. Estimate adoption costs and risks before adapting the DevOps
5. Make sure that good organisation and management structure is maintained
6. Provide a better understanding of business requirements by providing trainings
7. Recruit people for the project who have good knowledge about DevOps environment
8. Clarify issues that arise as soon as possible
9. Better to start DevOps in small projects

VIII. CONCLUSION

This paper identifies the tools that can be adopted in embedded system development where there is agile methodology in practice. We also discussed about how automation is done in embedded system, its benefits, challenges and solution to tackle difficulties. Initially there was limited number of tools supported for automation especially for the embedded systems. Jenkins was mostly used for web application development. But in this work Jenkins is made compatible for automation in embedded system where agile methodology is in practice. We found that this proposed methodology greatly improved the performance of CI/CD which made it more reliable when compared to previous methods.

References

[1] J. Humble, D. Farley, "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation", 1st ed. Boston: Addison-Wesley Professional, 2010

[2] D. G. Feitelson, E. Frachtenberg, K. L. Beck, "Development and Deployment at Facebook," IEEE Internet Computing, vol. 17, no. 4, pp. 8–17, 2013

[3] G. G. Claps, R. BerntssonSvensson, A. Aurum, "On the Journey to Continuous Deployment: Technical and Social Challenges Along the Way", Information and Software Technology, vol. 57, pp. 21–31, Jan. 2015.

[4] B. Fitzgerald, K.J. Stol, "Continuous Software Engineering and Beyond: Trends and Challenges", 1st International Workshop on Rapid Continuous Software Engineering, pp. 1–9, 2014.

[5] L. Crispin, J. Gregory, "Agile Testing: A Practical Guide for Testers and Agile Teams" Addison-Wesley, 2011.

[6] A. Askarunisa, K.A.J. Punitha, A.MAbirami, "Black Box Test Case Prioritization Techniques for Semantic Based Composite Web Services Using OWL-S", Proceedings of the IEEE-International Conference on Recent Trends in Information Technology, ICRTIT, 2011.

[7] E.M.A. Rauf, E.M. Reddy, "Software Test Automation: An Algorithm for Solving System Management Automation Problems", Proceedings of the International Conference on Information and Communication Technologies", ICICT 2014

[8] Patrick Day, "n-Tiered Test Automation Architecture for Agile Software Systems", Conference on Systems Engineering Research, 2014.

[9] B. P. Gopularam, Yogeesha C.B, P. Periasamy, "Highly Scalable Model for Tests Execution in Cloud Environments", 18th Annual International Conference on Advanced Computing and Communications (ADCOM), 2012.

[10] D. Jenkins, J. Arnaud, S. Thompson, M. Yau, and J. Wright, "Version Control and Patch Management of Protection and Automation Systems", 2013.

[11] Preeti Rai, Madhurima, S. Dhir, Madhulika, A. Garg, "A Prologue of JENKINS with Comparative Scrutiny of Various Software Integration Tools", 2nd International Conference on Computing for Sustainable Global Development, 2015.

[12] Yanmei Liu, Yanmei Lu, Yanmei Li, "An Android-Based Approach for automatic unit test",2015.

[13] Stephan Puri-Jobi, "Test Automation for NFC ICs using Jenkins and NUnit", IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2015

[14] M. Shahin, M. Ali Babar, and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices", IEEE Access, 2017

[15] L. Chen, ―Continuous Delivery: Huge Benefits, but Challenges Too, IEEE Software, vol. 32, no. 2, pp. 50-54, 2015

[16] E. Laukkanen, J. Itkonen, and C. Lassenius, "Problems, causes and solutions when adopting continuous delivery—A systematic literature review, Information and Software Technology", vol. 82, pp. 55-79, 2017

[17] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review, Journal of Systems and Software", vol. 119, pp. 87-108, 2016