Toward Unified DevOps Model

Abubaker Wahaballa,[a,b*], Osman Wahballa[a†], Majdi Abdellatief [§] , Hu Xiong[a‡], and Zhiguang Qin[a‡],
[a]University of Electronic Science and Technology of China, Chengdu, China
[*]wahaballah@hotmail.com [†]wahballa_777@hotmail.com [§]khwaja24@yahoo.com
[‡]{xionghu, qinzg}@uestc.edu.cn

***Abstract*—DevOps community advocates collaboration between development and operations staff during software deployment. However this collaboration may cause a conceptual deficit. This paper proposes a Unified DevOps Model (UDOM) in order to overcome the conceptual deficit. Firstly, the origin of conceptual deficit is discussed. Secondly, UDOM model is introduced that includes three sub-models: application and data model, workflow execution model and infrastructure model. UDOM model can help to scale down deployment time, mitigate risk, satisfy customer requirements, and improve productivity. Finally, this paper can be a roadmap for standardization DevOps terminologies, concepts, patterns, cultures, and tools.**

***Index Terms*—DevOps, conceptual deficit, Unified DevOps Model (UDOM)**

I. Introduction

IN the IT industry, development and operations often are in conflict during deployment. DevOps came to address this conflict and bridge the gap between developers and operations staff. The term DevOps is widely used these days, and many different types of content are associated with it. DevOps is a software development method that advocates communication, collaboration and integration between software developers and operations teams to solve critical issues, such as fear of change and risky deployment.

The time pressure often causes the conflicts between development and operations teams. Typically, a new software release must be deployed quickly. Another scenario in response, developers or operations team block communication and do not offer any help, or they do not react quickly when the system crashes. This situation often leads to a blame game where each side accuses the other of causing the problem. DevOps can help to avoid or mitigate the problems that our software industry faces today [1]–[3]. These problems include:

1) *Fear of change*: Once an application is delivered, the stakeholder tends to be tremendously afraid of change. This situation leads to bureaucratic management processes that makes any change to the application (be it the introduction of a new feature or a bug fix) take a long time to be effected.
2) *Risky deployments*: Software deployment is all of the activities that make a software system available for use. In this process, developers and operations teams are more worried, because no-one is quite confident that the software will work properly in its real environment. We just deploy it, and watch to see if it falls over.
3) *Blame game*: This problem is typically detected by Sysadmin or end-users. After investigation the problem is reported to the developers. This situation leads a classic blame-game scenario, then the *finger-pointing* may occur.
4) *Isolation*: In traditional settings, the project team is divided into two groups: a) Development team which includes programmers, tester, and Quality Assurance (QA). b) Operations team which include database administrators, system administrators, network administrators, and operators. Often, this division act like silos because they are independent of each other.

A. Motivations

The DevOps community fosters collaboration between development and operations staff; however this collaboration may cause a dangerous problem called a *conceptual deficit*. The Conceptual deficit comes from incomplete, wrong, or unimplemented nonfunctional requirements. On the other side, a DevOps advocates the automation to minimize cycle time, monitoring environment, and ultimately address errors by enabling testing more often, as automating incomplete or incorrect processes can be as disastrous as not automating at all. The motivations of this paper are twofold:

1) It discusses the origins of conceptual deficits;
2) A Unified DevOps Model (UDOM) is proposed in order to overcome these conceptual deficits that will eventually lead to a successful completion of automation works.

The UDOM model can help the organizations adapt to DevOps smoothly and easily, and that will maintain these organizations on the top of the IT competition roulette. Also this paper can be starting-point for standardization DevOps terminology, concepts, patterns, cultures, and tools.
The paper is organized as follows: In the next section, the origins of conceptual deficits are discussed. In Section III, we propose a unified DevOps model. We finally conclude the paper in Section IV.

II. The origins of conceptual deficits

To improve the process of software and deliver product value on continuous basis, we need the full collaboration between the developers and operations team. However this collaboration may cause conceptual deficit problem. In this section we will discuss the origins of this problem and how to avoid or mitigate it. The conceptual deficit may come from: *i*) unimplemented nonfunctional requirements *ii*) bounded rationality *iii*) complex and dynamic environments *iv*) the principal-agent problem *v*) moral hazard.

978-1-4799-8353-7 /15/$31.00 ©2015 IEEE

A. Nonfunctional Requirements

The FURPS model [4] is proposed for unifying and classifying the software quality attributes (functional and non-functional requirements). FURPS stands for: Functionality, Usability, Reliability, Performance and Supportability. Developers must consider and define the non-functional requirements very well in the early stages, otherwise it become very complex and expensive to address them later on.

B. Bounded Rationality

Many economics models [5] assume that people are on average rational, and can in large enough quantities be approximated to act according to their preferences. The challenge of bounded rationality in trying to build models (e.g., machines) that generally predict the decisions that we make. Rational analysis institutional contexts [6] and computational intelligence [7] could play a big part to overcome this challenge.

C. complex and dynamic environments

Today's IT industry is confronting by a number of complexities:

1) Dynamic and heterogeneous middleware environments, for example cloud and virtual environments;
2) The increased automation of nearly all business functions to minimize cycle time, monitoring environment, and ultimately address errors;
3) adoption to DevOps and service-oriented architectures (SOA).

This makes it much more difficult for most organizations to manage their software applications. Thus, awareness and commitment of management to DevOps are necessary.

D. The principal-agent problem

The problem of motivating one party (the agent) to act on behalf of another (the principal) rather than in his own interests. Principal-agent relationships should reflect efficient organization of information and risk-bearing costs to avoid the conflict between them.

E. Moral hazard

Moral hazard [8] occurs when a decision-maker insulated from the consequences of his or her decision. However, If consultants, or IT managers do not bear the full consequences of their actions, they may not put in place the appropriate measures needed when the errors occur or the system crashes.

III. Proposed Model: A Unified DevOps Model (UDOM)

In this section we proposed a Unified DevOps Model (UDOM) to unify DevOps processes, terminology, concepts, patterns, cultures, and tools. UDOM includes three sub-models that are:

1) Application and data model - '*what*'
2) Workflow execution model - '*how*'
3) Infrastructure model - '*where*'

A. Application and Data model

Application data modeling is a collection of the techniques and tools used to translate the complex system designs into easily understood representations of the data flows and processes. The proposed application model is expressed diagrammatically in Figure 1. It allows the developers and operators to define the functional and nonfunctional requirements collaboratively according to FURPS model. When nonfunctional issues are addressed in the early stages, developers can write code to meet those requirements. It actually reduced the cost.

In the plan, developers review the commitments and leverage information within organization then they estimate the resources, effort, and schedule with incorporating the knowledge of individuals / groups in the decision making process and gaining their support for the commitments made.

In development phase, developers concentrate on new features to be developed and write source for solutions that implements these features. Then collaboratively, other developer '*peer reviews*' the code to find and fix mistakes overlooked in the early stages to avoid problems later on. Peer review [9] is systematic examination of code to find and fix errors overlooked in the early stages, improving both the overall quality of software and the developers' skills.

Testing is a significant turning point between development and IT operations, and it is important to start it as early as possible. In this phase tester will typically need to:

- *Setup testing environment*: build the machinery, the services, virtualization and test lab management tools;
- *Identify testing strategy*: this includes the testing objective, methods of testing new functions, roles and responsibilities, total time and resources required for the project;
- *Fast feedback*: rapid and continuous feedback allow the team forward continuously and assist in test the project quickly.

Releases are planed as follow:

- Management all releases in a spreadsheet updated through face-to-face meetings
- Definition of policies for differing release types (i.e. Major, Minor, Emergency).
- Provisioning Infrastructure and middleware similarly.
- Definition of workflow and approvals for moving releases through environments.
- Planning and publication of release content, dates and success criteria.
- Analyzing the impact of change among releases;
- Identify the relationship between releases of multiple applications.

B. Workflow execution model

Workflow execution model provides how to configure and execute workflow elements [10]. lt help the IT companies to organize their work flow and to monitor the progress, as well as it keeps accurate timesheet records to record the start and end time of tasks. Our workflow model is divided into four quarters as shown in Figure 2. Each quarter includes:

- Workflow purpose;

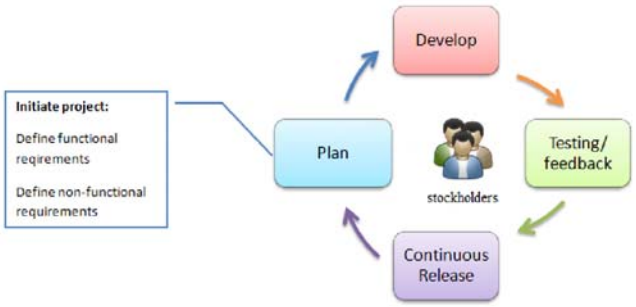

Fig. 1. Proposed Application Model .

- Workflow steps;
- Workflow step evaluation process;
- Workflow participants.

1) First Quarter- Software Plan workflows: The purpose of this workflow is review the commitments and leverage information within organization, and then estimate the resources, effort, and schedule. The participants of this workflow are: developers, operations team, and stockholders; everyone can equally contribute. The workflow steps include:

- Reviewed the commitments and leverage information within organization;
- Estimation the resources, effort, and schedule;
- Evaluation the plan formally with feasibility and acceptability to stakeholders;
- Bases for the fine-grained plan for the next iteration (the iteration plan).

2) Second Quarter- Software development workflows: The purpose of this workflow detail is to create the artifacts and enclosures of the Software components. The participants of this workflow are developers and operation team, where they defined the functional and non-functional requirements collaboratively according to FURPS model and then the other workflow steps are implemented as follow:

- Concentration on new features to be developed;
- Write the source solution that implements these features;
- Review the code to find and fix mistakes collaboratively by other developer (peer-reviewer).

3) Third Quarter - Testing: The purpose of this workflow is to verify and validate that a software application meets the business requirements that identified in early stages. The participants of this work flow are: Testers, QA, developers, and stockholders. The workflow steps include:

- Setup the test environments;
- Identify testing strategy;
- Detection the flaws, and errors in the application code that must be fixed;
- Send continuous feedback.

The evaluation of this workflow is conducted cooperatively with stakeholders to verify and validate the quality of the product or service under test.

4) Fourth Quarter - Continuous Release: The goals of this workflow detail is to plan, manage, execute and control the release schedule and track a release through every stage of the delivery life-cycle to ensure production worthiness. Each one in work team is responsible for the delivery process. The workflow steps include:

- Identifying release dependencies: Form start-to-end, easily to identify and track a release dependencies (users' stories) that either has a predecessor or a successor;
- Identifying release plan and track: Map applications, assign features to releases, break down features into stories, assign stories to iterations and start monitoring progress.
- Stable-Stable easier release days: Build an executable deployment plan that smoothly notifies the team of the work to do, and triggers automated tasks while indicating to current status that everyone can see.
- Accommodate more releases: Set-up more templates visibility with auto-rules to accommodate a growing number of release that mitigates risk.
- Multi-Application Continuous Delivery: Identify automatically the relationship between releases of multiple applications.

C. Infrastructure Model

Our infrastructure model aims to simplify and provide the full control over infrastructure. It builds based on Infrastructure Optimization model [11] and Infrastructure Maturity Model. As shown in Figure 3, infrastructure model includes six optimization levels: Basic, standardized, rationalized, virtualized, service-based and code-based.

1) *Basic IT Infrastructure*: This level is characterized by manual, localized business processes; minimum of central control; complex management, and nonexistent or unenforced IT policies standards for security, backup, deployment, compliance, and other common IT requirements. In this level, the detailed information about infrastructure and platform is not available and therefor, it is impossible to follow any tactics for the purpose of improvement.
2) *Standardized IT Infrastructure*: This level provides controls through the use of standards and policies to manage desktops and servers. For example, Microsoft's Infrastructure Optimization Model allows for the management of the network resources, security policies, and access control using Active Directory Domain Services (AD DS).
3) *Rationalized IT Infrastructure*: This level aims to rationalize the costs involved in managing desktops and servers are at their lowest and where processes and policies have matured to begin playing a large role in supporting and expanding the business. Security and safety policies are now proactive and responding to threats and risks are rapid and controlled.
4) *Virtualized IT Infrastructure* This level aims to reduce the effort and cost of deploying and maintaining desktops and applications and resolve application compatibility issues that can make traditional deployment difficult and time-consuming .

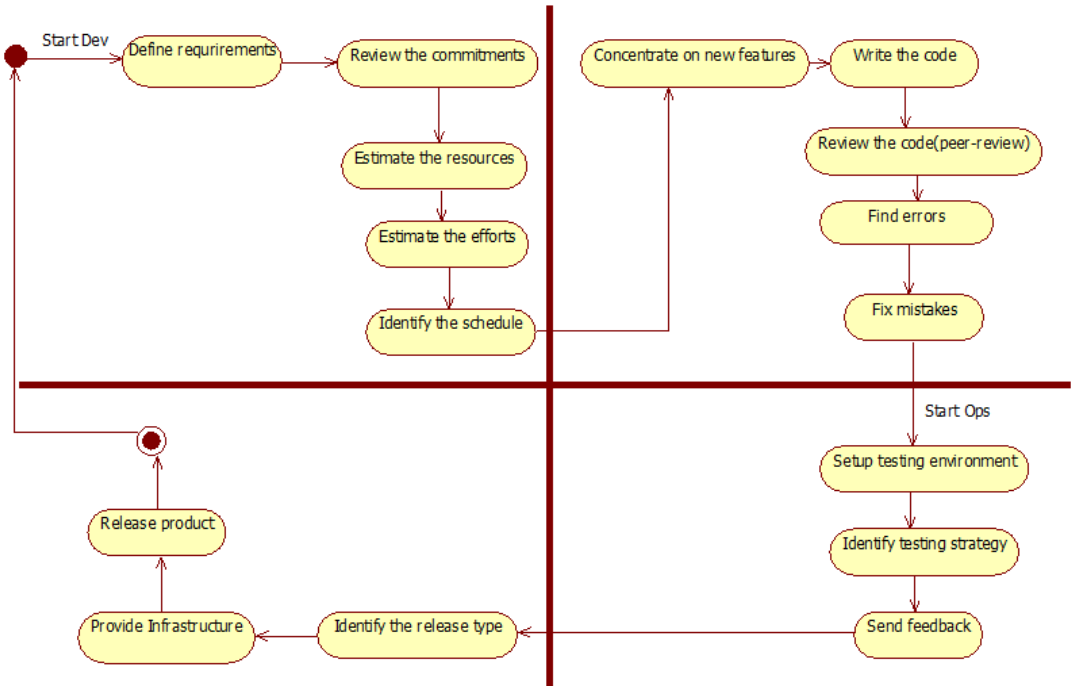

Fig. 2. Proposed Workflow Model.

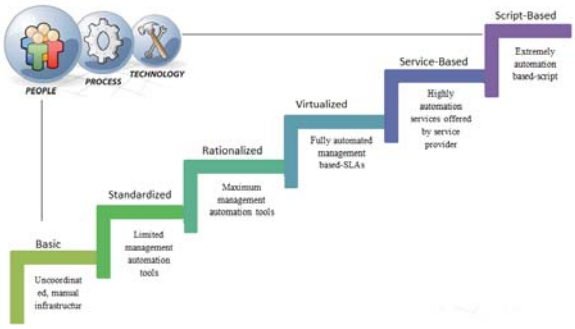

Fig. 3. Infrastructure Model.

5) *Infrastructure as a Service (IaaS)*: This level offers high automation control, where an organization outsources the equipment used to support operations, including storage, hardware, servers and networking components are owned and hosted by a service provider. The client typically pays on a per-use basis.
6) *Based-Script IT Infrastructure*: This level introduces extreme control based-script environments from start-to-end, through installing an operating system, installing and configuring servers on instances, configuring how the instances and software communicate with one another, and much more. By scripting infrastructure, it would have full control to scale up and down without any additional effort.

IV. Conclusion

Due to the rapid development and high competitiveness in IT industry, software developers are pressured to quickly complete and release new code. On the other side, operators aim to update and keep production systems stable at all times. This paper proposes a Unified DevOps Model (UDOM) in order to overcome the development-operations barrier. UDOM includes three sub-models: application and data model, workflow execution model and infrastructure model. UDOM unifies DevOps processes, terminology, concepts, patterns, cultures, and tools.

References

[1] P. Debois, *Agile Infrastructure and Operations: How Infra-gile are You?*, Agile, 2008. AGILE '08. Conference, pp.202,207, 4-8 Aug. 2008
[2] Wettinger, Johannes; Breitenbcher, Uwe; Leymann, Frank: Standards-based DevOps Automation and Integration Using TOSCA, International Conference on Utility and Cloud Computing, 2014.
[3] P. Duvall, *Agile DevOps: Infrastructure automation*, Copyright IBM Corporation, 2012.
[4] M. Umar, N. Khan, *Analyzing Non-Functional Requirements (NFRs) for software development*, Software Engineering and Service Science (ICSESS),IEEE 2nd International Conference ,Beijing, pp. 675-678, 2011.
[5] A. Rubinstein. *Modeling Bounded Rationality*, MIT Press, USA, 1998.
[6] Tan, J. K., Tan, E. L., Kanagalingan, D. and Tan, L. K. (2014), Rational dissection of a high institutional cesarean section rate: An analysis using the Robson Ten Group Classification System. Journal of Obstetrics and Gynaecology Research. doi: 10.1111/jog.12608.
[7] T. Edward, *Computational Intelligence Determines Effective Rationality*, International Journal of Automation and Computing, pp. 63-66, 2008.
[8] Dembe and Leslie I. Boden, *Moral Hazard: A Question of Morality*, New Solutions10, 3, pp. 257-279, 2000.
[9] D.Trytten, *A Design for Team Peer Code Review*, Proceedings of the 36th SigCSE Technical Symposium on Computer Science Education,v.37 n.1, 2005.
[10] S. T. Maine, J. D. Brown, and E. Pinto. "Workflow execution model.", U.S. Patent Application 13/922,197, filed June 19, 2013.
[11] B. Curtis, B. Hefley, S. Miller, *People Capability Maturity Model (P-CMM) Version 2.0*, Second Edition, Software Engineering Instiute (SEI), 2009.