IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 2, Ver. IV (Mar – Apr. 2015), PP 51-57
www.iosrjournals.org

DevOps shifting software engineering strategy Value based perspective

Samer I. Mohamed
[1](*Electrical and Communication Engineering, Faculty of engineering/ October University for Modern Sciences and Arts, Egypt)*

Abstract: *Distributed software engineering is one of the hot research areas that has the most interest within the software industry for many IT organizations especially the global and multinational ones. Up to 90% of these organizations running projects are distributed over different teams and even countries, which puts Global Software Engineering (GSE) with high value from both industrial and academic perspectives. Many challenges attack these organizations due to current economical slow down makes delivery efficiency, cost control and time to market key Critical Success Factors (CSF) to survive in these market conditions. DevOps is newly emerging metrology stresses communication, collaboration and integration between software developers and technical operations team to rapidly deliver software products and services to the market. The value behind DevOps is two folds: first one is mitigating the challenges faced by distributed software engineering and second is to bridge any gaps within current traditional organization processes. This paper introduces a new DevOps maturity model and assesses how this model will impact existing GSE practices and processes.*

Keywords: *Critical Success Factors - Collaboration -Continuous delivery - Continuous deployment – Continuous Integration Distributed development – DevOps - Governance - Global Software Engineering - IT operations – Quality.*

I. Introduction

Global software engineering refers to the software engineering executed with globally distributed settings and practices to satisfy the emerging needs from the software industry and software organizations on a standard and best practices that help them fulfill their clients demand. The software delivery life cycle - starting from Proof of Concept (PoC) till deployment via design, development, testing, and integration - can be implemented either by different teams within the same site/location, multi-site within same country or even on multi-national basis. Each team, department, site or country combines its competencies, skills, experts, technological edge and/or talents to maximize the value gain towards the organization clients while maintaining/managing their respective operational risk and costs [1].

GSE has many potential benefits where global and multi-national organization gains in addition to improving their time-to-market by following 'follow the sun' theme and/or operational margin through outsourcing approach. One of these benefits is innovation results from wide spread/access of these organizations to different software engineers from different countries/cultures with different skills, talents, technological backgrounds, competencies, and ideas. Without these potential benefits, organizations were not able to invest such billions.

There are intrinsic risks/challenges around distributed nature of software engineering like cultural differences, communication, team dispersion, time zone differences, lack of trust between different teams/partners, unclear assignments, roles and responsibilities of each role within the project, quality issues, project delivery failures, coordination conflicts, cohesion barriers, and/or Intellectual Property Rights IPR management. These sample of challenges/risks have direct correlation with the final software product delivery towards the client consequently organization revenue and/or margin. This is basically why these organizations invest a lot to improve their processes to adapt always with their client needs, product deliverables, labor issues and tools [2].

DevOps is an evolution in thinking with regards how IT services are delivered and supported. It is a continuation of some of the predecessor work in the areas of continuous integration and application life cycle management (ALM); therefore, it is rooted in the agile philosophy, which also attempts to bridge the traditional organizational process divide between development and operations teams [11].

The value behind DevOps lays in bridging the current gap between the different technical roles within the same team who work in silos. Thus, The DevOps approach is built around those who believe that the application of a combination of appropriate technology and attitude can revolutionize the world of software development and delivery especially these different roles share the same objective which is the delivery of a successful products under a stressful market conditions [3].

The paper is organized as follows: section II gives a background for the early studies done on global software engineering and DevOps as a concept; section III provides the main challenges that tackle DevOps and how the emerging methodology overcomes and prevails its objectives; section IV introduces a new DevOps maturity model based on CMMI maturity mode; section V proposes a new transformation framework towards DevOps work structure using the proposed DevOps maturity model; section VI is the conclusion for what has been presented in the paper.

II. Background

GSE challenges are discussed in many publications from various perspectives. The research done within [4] carried out a systematic literature review of distributed development challenges, best practices, models and tools. They analyzed 54 papers and found that the top-five challenges appeared in 120 pieces of evidence (45%) out of a total of 266 for all 30 identified challenges. These five challenges were effective communication, cultural differences, coordination, time-zone differences and trust. These challenges affect all aspects of product development and different authors have studied these aspects in more detail either from certain process viewpoints or from the challenge viewpoint.

The most critical points in global software engineering, based on the industrial inventory and expressed by the Merlin and Prisma partners in workshops [5], were the contracting and requirements definition, project planning and tracking, architecture analysis, design and integration. Referring to Contracting and requirements definition, the more detailed the prepared specification of the work is, the better (within a reasonable degree of effort). Thus, if all collaboration partners have the same view/shared understanding of what is to be done and if that is documented well, fewer conflicts will occur. Project planning and tracking; It is important to clearly define status-reporting practices and change management procedures, including the details on reporting channels, decision authorities and escalation channels. Architecture analysis/design; Architecture is one of the key disciplines enabling successful collaboration. In particular, a lack of sound architecture leads to poor integrality. Ensuring that all partners understand the architecture correctly is difficult and the required level of communication is often underestimated. Integration and testing; While integration is often the most time and effort consuming activity even in in-house product development, GSE brings additional complexity; for example, new actors and communication requirements. Co-operative work: effective, timely and accurate communication is regarded as a CSF for projects. Openness of communication and multisite/multi-partner culture were considered as very important for collaboration success.

Adopting the traditional concepts and methodologies within software engineering in general and GSE in specific shows that for any artifact to be promoted to live environment takes on average more than 1 day [6]. This proves to be out of sync with the current market needs where time to market is one of the major CSFs for any organization. Thus a need for new approach raise to bridge this gap between current software engineering practice and market needs. Currently we live in the era of the application, a place where business agility depends on application agility. This makes the rise of agile methods over traditional ones. Sequential application delivery one of the most compelling modernization stories of the past decade. And because it encourages collaboration between application and business teams, it also promises software more closely aligned with business requirements.

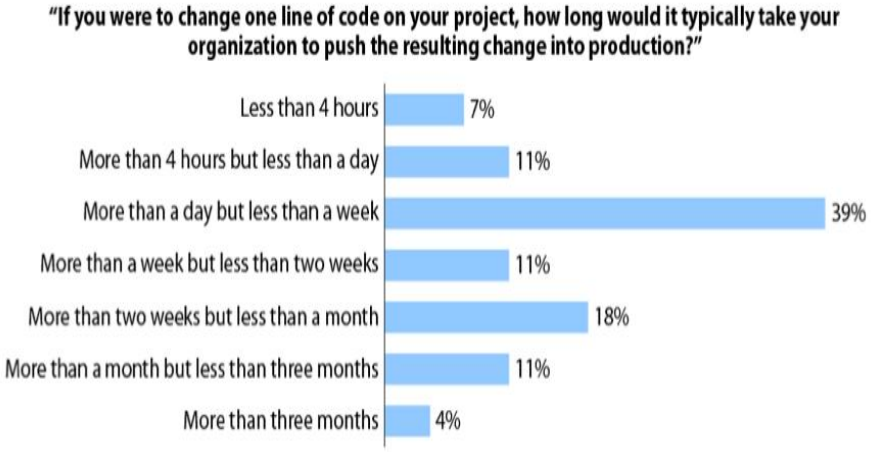

Fig. 1. Traditional release process statistical view for any change

III. DEVops challenges

The emerging DevOps movement takes aim to overcome the gap between market needs and traditional software engineering approaches by seeking to close the gap between Development and IT Operations. DevOps is a set of best practices and methods inspired by Agile that focuses on better collaboration between the two groups. Continuous Delivery (CD), which is enabled by DevOps, focuses on what is most important: shorter cycles for actually putting functionality in the hands of users. It relies not only on better collaboration, but on comprehensive automation of the build, test, and deployment process [7].

This level of integration between development and operations becomes revolutionary because it enables releases to be driven by business need as opposed to operational constraints. The keys to successful adoption of DevOps and Continuous Delivery are quality, automation, collaboration, and governance/process. Together, these fundamental elements can unify the traditional IT silos to enable agility across the end-to-end application life cycle.

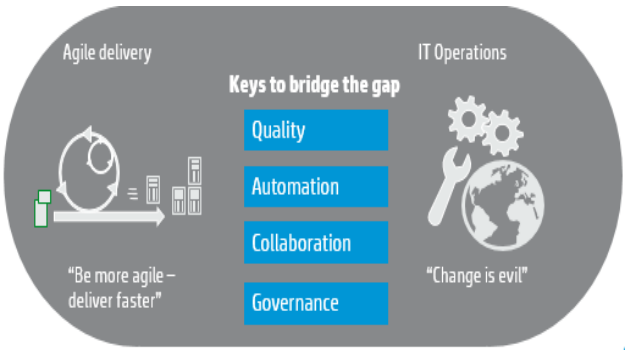

Fig. 2. DevOps delivery model

DevOps ensures to bridge the current gap between development and IT operations arms of any team/organization via group of best practices/processes on different layers like Quality, Automation, Collaboration and Governance as shown in Fig 2. Quality layer ensures to deliver in more lean/faster manner towards live environment facilitate limiting the rework time, make application more stable and produce releases with higher quality targets. Automation layer improves delivery speed, throughput/productivity, and repeatability. Collaboration layer ensures better communication between different teams within same project via different communication approaches/tools to minimize risk, rework cost and maximize value towards client as teams will be in touch from day one and not wait till delivery to communicate [8]. Governance layer is primary responsible to control how these layers work seamless together to ensure better achieving of the global objective from introducing DevOps delivery model.

IV. Devops Maturity Model

The proposed maturity model is based on CMMI maturity model [10] with five levels of maturity. Each level is assessed against 4 dimensions (Quality, Automation, communication/collaboration, and governance) as described in DevOps delivery model Fig 3. To move from one maturity level to the following one, organization needs to improve the 4 dimensions. This proposed model is inherited from HP model [9] to cover the entire life cycle of the application or delivered service.

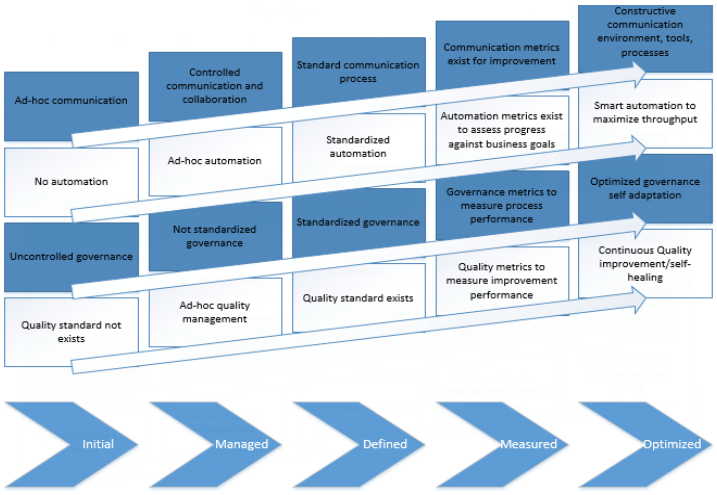

Fig. 3. DevOps maturity model

Maturity level 1: Initial

- Communication maturity: Communication normally within this maturity level is done on ad-hoc basis with no clear process or tool. No clear roles and responsibilities between different team members within same team that lead to chaotic environment and communication overhead. Decision making is mostly centralized by the process owner with no clear reference/basis then communicated to teams.
- Automation maturity: Normally there is no automation implemented at this maturity level either for services or activities and most of the tasks are done manually with no automation.
- Governance maturity: Governance/process is basically ad-hoc at this maturity level where outcomes of any process/service are not predictable. For that there are no control measures in place to manage the inputs/outputs of any process/service.
- Quality maturity: Quality has no significance or value and mostly done on ad hoc basis based on the process owner initiative/objectives.

Maturity level 2: Managed

- **Communication maturity**: Communication is much more coordinated/managed between different stakeholders. Although decision making is still centralized, decisions are shared between teams in much more concrete/managed approach. Roles and responsibilities are more defined and well shared between different stakeholders that lead to minimized communication overhead and better achieving of the communication objectives. Communications at this level are not yet shared between teams and accountability is missing lead to duplications and extra costs/time to market.
- **Automation maturity**: Automation process is documented but not yet executed as a standard. It's basically done on ad-hoc basis within the organization according to each project champ objectives.
- **Governance maturity**: Governance is executed at this maturity level on ad-hoc basis and not yet becomes a standard within the whole organization. Even the processes/procedures within the different projects may be different to fulfill specific and customized project need or requirement.
- **Quality maturity**: Quality starts to have a value at this level and managed by each project on ad-hoc basis but not yet become a standard on the organization level. Defect tracking/management is done using proper tools but based on project need and not based on clear organization requirement or standard.

Maturity level 3: Defined

- **Communication maturity**: Communication starts to be more concrete between the different teams even if they are co-located or distributed on different locations/countries using the proper tools/mechanisms. Communication management becomes essential and even part of organization governance/process standard itself. Teams are more aligned ahead, for example testing/quality team is connected with Development team early in the development cycle to create the required test cases for each new module/component. Besides IT operations team is also aligned with development team before the deployment around the defects type and how to prevent them. Through this communication, tools, ideas sharing, teams will be able to remove silos and avoid duplications. This will remove any time/tasks loses, the thing which improves the ROI and time to market.
- **Automation maturity**: Automation at this maturity level is adopted as a standard from the organization itself and not done as silos like what happen on the managed maturity level. Through this each organization provides the proper automation framework, infrastructure, technical training, tools and processes/governance to facilitate better outcomes from this stage towards the optimum goal/objective of maximizing the organization throughput/productivity.
- **Governance maturity**: Governance and processes at this maturity level become a clear organization standard, adopted by the whole organizational and managed by dedicated team who is responsible for mandating this over the different running projects in a consistent manner. Each project team still have the ability to tailor the standard process/governance based on its special need/requirement while still keep fitting the whole/generic organization process framework.
- **Quality maturity**: Quality at this maturity level us adopted as a standard on the organization level where quality processes and tools adopted in consistent manner by different project teams. Defect tracking and touch points between testing/quality team and development teams in the development life cycle are defined with clear toll gates for each phase. Regardless the development methodology either waterfall or agile, the quality team is following the organization standard to achieve the mandated goals.

Maturity level 4: Measured

- **Communication maturity:** At this maturity level, communication metrics and analytics are calculated/measured to identify areas of improvements and tackle any gaps within current communication/collaboration process. Metrics like types of communication media and how it fits for purpose and

fits for use, and frequency of alignments between different teams against deliverables from each project gate are sample of these metrics.

- **Automation maturity**: Automation measurements at this maturity level is vital to show the value behind the automation and if it still worth to continue investing in this area against business needs. Metrics like ROI, time to complete, incurred budget, involved stakeholders against each business goal are sample of the collected data to provide more predictability/visibility towards organization upper management.
- **Governance maturity**: Without measuring the existing process/governance, nothing will be improved. Thus at this maturity level, the performance of the governance processes themselves are measured using different statistical historical data/techniques from respective domain/application. Collecting data related to number of deployments, E2E release cycle efficiency,
- **Quality maturity**: Quality measurements are very important and essential to deliver competing products in stressful market that are able to survive in current harsh market conditions. To be able to deliver such high quality products, quality metrics like total rework costs, number of defects, number of incidents, Service Level Agreements (SLA) compliance, number of breaches, test coverage, number of rollbacks, build/deployment cycle time, time to market are very important to evaluate product quality levels and draft the road map for the corrective actions that need to be taken to revert back to the main organization goals/objectives.

Maturity level 5: Optimized

- **Communication maturity**: Main objective at this level is to better optimize the communication and collaboration processes between different teams to better improve and to ensure continuous knowledge sharing and adaptation to any dynamics that affect current organization objectives.
- **Automation maturity**: Analyzing the outcomes from the automation metrics measurements to identify the areas where more automation is important compared to others and to align better with the organization goals and objectives. More focus towards smart or self-adaptive automation techniques at this level will better align the organization time to market goals.
- **Governance maturity**: Focus at this maturity level to better optimize the governance process based on the collected measurements/metrics to better align towards organization goals and market dynamics.
- **Quality maturity**: Utilize the collected quality metrics to build improvement plans to enhance the product quality in alignment with the organization objectives.

V. DEVops transformation framework

To enable organizations get the most outcomes/value from the DevOps, a transformation framework is proposed to first assess the current state of the organization/department based on the DevOps maturity model described in the previous section, and then apply the proposed transformation framework. This transformation framework will apply set of processes and best practices to make the organization work model adaptable with the DevOps model. Transformation framework consists of five phases as described in Fig 4. The transformation model phases are as follows:

- **Initial assessment phase**

The main objective at this phase is to do a gap analysis and assessment for the current state (AS IS). It will list as well the main pain areas which affect organization/department throughput. Deliverables from this phase will be gap analysis, current state identified by (processes, tools, and work structure).

- **Roadmap identification phase**

The main objective from this phase is to draft the roadmap/planning for the whole transformation project. The deliverables from this phase will be like, which team to start with, set of processes to be modified, set of tools to be acquired/adapted to facilitate the transformation, quick wins, and collaboration/communication tools.

Fig. 4. DevOps tranformation framwwork

- **Transformation execution phase**

Real implementation starts at this phase, where transition from current state (AS IS) of the pilot organization/department towards the target state (TO BE) is implemented. Deliverables from this phase will be like, the new DevOps model identified in terms of the new set of process/governance rules, automation tools/processes, communication processes between different teams, and quality control processes/tools to better ensure continuous delivery and continuous deployment towards production environment.

- **Verification phase**

The main goal of this phase is to verify the outcomes/deliverables End to End (E2E) for the whole development life cycle using the DevOps new transformed model against old model. This will include mercies collection like (Time to market, Number and frequency of software releases, Defect escape ration, time/cost per release, Mean Time To Recovery (MTTR), environment down time, number of deployments along with deployment time, and team productivity).

- **Optimization phase**

The objective from this phase is to ensure continuous improvement and optimization for the whole DevOps service/model on the organization level to ensure better orchestration/harmony between the different teams towards optimum organization goals/objectives where lean mindset spread over to better focus on customer need, doing things right at first time, continuous improvement and ability to 'stop the line' to fix issues immediately when they occur.

VI. Conclusions

DevOps model proves not to be just an emerging movement that advocates the collaborative working relationship between development and IT operations but also helps in shifting the current software engineering practices and processes resulting in fast flow of planned work while simultaneously increasing the reliability, stability, resilience and security of the production environment. Through this paper we introduced a new DevOps maturity model based on CMMI maturity model to enable assessing any organization working model/state against DevOps model. The paper introduces as well a transformation model that helps in transforming the software development life cycle to adapt with the DevOps strategy. This leads to accelerate the delivery of application changes to production environment, improve the operational efficiency with the implementation of consistent and standardize deployment model, mitigate the risk of downtime due to deployment errors, and better visibility and traceability to meet audit and compliance objectives.

References

[1]. Juristo, N., Moreno, A.M., and Silva, A.A. 2002. Is the European Solving Industry Moving Toward Requirements Engineering Problems? IEEE Software 19(6): 70-77

[2]. Komi-Sirviö, S, and Tihinen, M. 2003. Great Challenges and Opportunities of Distributed Software Development - An Industrial Survey. In proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering, SEKE2003, San Francisco, USA pp. 489 – 496

[3]. Jez Humble, Chris Read, Dan North, The Deployment Production Line, Proceedings of Agile 2006, IEEE Computer Society

[4]. da Silva, F.Q.B., Costa, C., Frana, A.C.C. & Prikladinicki, R. 2010. Challenges and Brazil, solutions in distributed software development project management: A systematic literature review. In: Global Software Engineering (ICGSE) 2010 5th IEEE International Conference on Global Software Engineering, Recife, pp. 87–96

[5]. 2009–2011. ITEA2 project. Productivity in Collaborative Systems Development URL: http://www.prisma-itea.org (Accessed 19 March 2012).
[6]. Carmel, E. 1999. Global software teams: Collaborating across borders and time zones. Prentice-Hall, Upper Saddle River, N.J. ISBN-13: 978-0139242182
[7]. Jez Humble and David Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley, 2010
[8]. LIVENSON, I., SINGER, G., SRIRAMA, S. N., NORBISRATH, U., AND DUMAS, M. Towards a model for cloud computing cost estimation with reserved resources. In Proceeding of 2nd International ICST Conference on Cloud Computing, CloudComp 2010 (2010).
[9]. http://h30499.www3.hp.com/t5/Business-Service-Management- BAC/DevOps-and-OpsDev-How-Maturity-Model-Works/ba-p/6042901#.VNgkEKP8IiQ (Available Jan 2015)
[10]. SACKS, M. DevOps principles for successful web sites. In Pro Website Development and Operations. Springer, 2012.
[11]. P. Debois. Opening statement. Cutter IT Journal,24(8):3{5, 2011.
[12]. B. Keyworth. Where is it operations within DevOps?Cutter IT Journal, 24(12):12{17, 2011