DevOps Maturity Calculator DOMC - Value oriented approach

Samer I. Mohamed

Department of ECE, October University for Modern Sciences and Arts (MSA), Cairo, Egypt

Abstract— *New style of IT puts many organizations in the current IT market under pressure to move their traditional delivery approaches towards modern mechanisms like DevOps. These new modern approaches consider the high demand rate of their clients who starve for having new features deployed to production every min to satisfy their needs especially with cloud computing and virtualizations become very common. Building on that needs, those organizations need to develop on their DevOps capabilities and invest on their resources skill set to cope with this new style of IT. Objective of this research work is to help those organizations to develop their DevOps capabilities and process maturity via designing a quantitative framework/tool that calculates the DevOps maturity level of the respective organization or project. This research is based on the work introduced in [Samer I. Mohamed] that describes the DevOps transformation framework. The proposed tool called DevOps Maturity Calculator or DOMC which measures the organization or project maturity level based on its level of maturity corresponds to each and every DevOps capability within the transformation framework introduced in [Samer I. Mohamed] DOMC introduces new set of features that enable organization to promote their business process by embed the DevOps maturity framework model along with set of metrics (KPIs, CSFs) to measure the current and expected mode of operation based on the organization targets/objectives..*

Keywords— Metric based, Transformation framework, Key Performance Indicator, Critical Success Factor, Maturity model.

I. INTRODUCTION

New style of IT is the main slogan adopted by many IT organizations currently to shift their industry stack to satisfy and meet their increasing clients' needs/demand. The new style of IT is basically characterized by cloud computing, security, big data and mobility which constitute the main four legs of any successful IT business and drive the engine of business growth. Starting with this vision in mind, many organizations start to shift their compass and begin sever business transformation towards the new style of IT strategies and work processes. One of these work process is DevOps strategy. DevOps is a new term emerging from the collision of two major related trends. The first called "agile system administration" or "agile operations"; it sprang from applying newer Agile and Lean approaches to operations work. The second is a much expanded understanding of the value of collaboration between development and operations staff throughout all stages of the development lifecycle when creating and operating a service, and how important operations has become in our increasingly service-oriented world.

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 [10].

DevOps emerged from a "perfect storm" of these things coming together. The growing automation and tool chain approach fed by more good monitoring and provisioning tools, the need for agile processes and dev/ops collaboration along with the failure of big/heavy implementations of ITSM/ITIL – they collided and unconsciously brought together all three layers of what you need for the agile movement (principles, process, and practices). Since then it has developed, most notably by the inclusion of Lean principles by many of the thought leaders.

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 [17].

One of the major challenges that face many It organizations is the balance between the change management and the continuous delivery. The objective of change management is to ensure standardized methods and procedures are used for efficient and prompt handling of all changes to control IT infrastructure to minimize the number and impact of related incidents upon

service. Changes in IT infrastructure may arise reactively in response to problems or externally imposed requirements, e.g. legislative changes, or proactively from seeking improved efficiency and effectiveness, or to enable or reflect business initiatives, or from programs, projects or service improvement initiatives. While the objective of continuous delivery is to satisfy the increasing demand from the business no matter what the implied processes. DevOps is the key to enable organization to overcome and mitigate this challenge by forcing the processes aspect via maturity levels/model [25] to meet the change management requirements while facilitate via tools/practices/capabilities between the different involved teams (development, testing, Quality assurance, operations) towards end to end delivery.

The paper is organized as follows: section II gives a background for the DevOps maturity model where the transformation framework is based; section III provides the main construction of the proposed transformation framework; section IV Validate the new DevOps transformation framework via use case; section V is the conclusion for what has been presented in the paper [12].

II. DEVOPS MATURITY MODEL BACKGROUND

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, and then apply the proposed transformation framework according to the gap assessment results. This transformation framework will apply set of processes and best practices to make the organization operating model adaptable with the DevOps model. The proposed model is based on the maturity model introduced in [23] with five levels of maturity. Each level is assessed against 4 dimensions (Quality, Automation, communication/collaboration, and governance) as described in DevOps maturity model in Fig 1. To move from one any maturity level to the following one, organization needs to improve against the 4 dimensions. This proposed model is inherited from HP model [30] to cover the entire life cycle of the any delivered service E2E (End to End).

At the **initial level of DevOps maturity**, organizations are capable of doing ad hoc deployments of services when they are ready, but they have a hard time predicting when those services will be production quality and are dependent on the actions of talented resources. At that level of maturity, there are many manual steps involved in deploying a new version of software service/application. As a result, a full service release cycle (inclusive of testing) can take days or even weeks to complete. The results of maintaining an initial level of maturity are that it's difficult for business leaders to predict when new software services will be released to their clients, and the teams that construct these services will usually deliver them later than predicted or expected from the clients. Clients will also frequently encounter regressions in functionality or system errors caused by mistakes in manual processes. As a result, organizations with an initial level of DevOps maturity will find their capacity to innovate through custom software severely constrained [3].

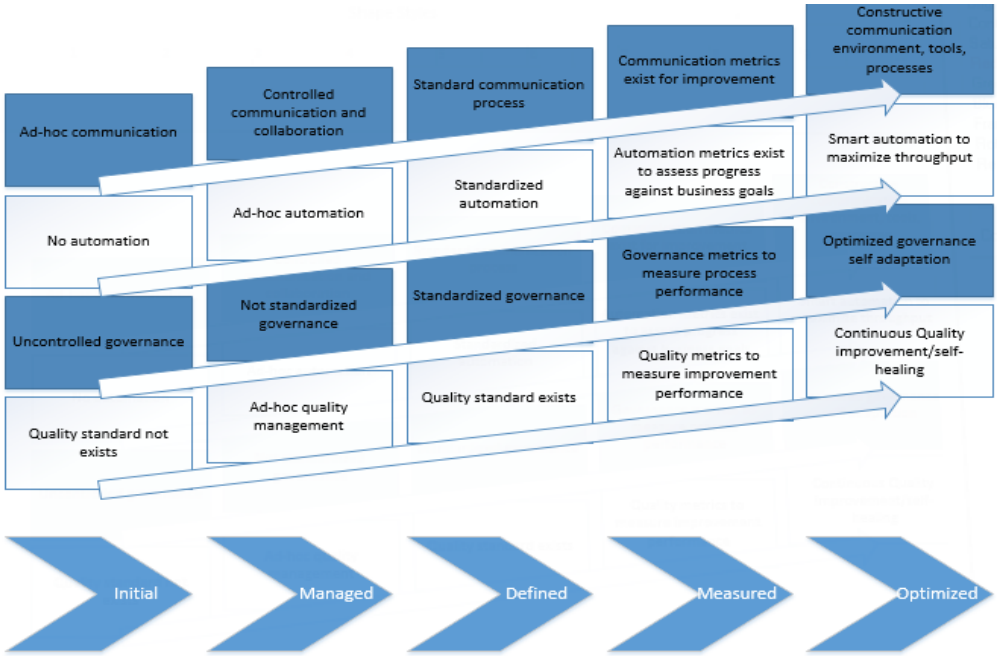

FIG 1. DEVOPS MATURITY MODEL

At the **managed level of DevOps maturity**, software development teams are able to work with business leaders to set release time boxes. Software development teams then vary the scope of the work performed and the effort applied to meet the agreed release date. At the managed level, delivery time boxes are usually still outside the bounds of what the business may need. As a result, a managed level of DevOps maturity is characterized by frequent negotiations over the priority of requirements, defects, and system capabilities versus budget and the release window. Because this negotiation process continues throughout the release, it's critical to have an identified product owner who can examine functionality and quality trade-offs and maintain customer support for the service while it's being developed. At this level, business sponsors regularly participate in scope/resource/date trade-offs and are well informed about the progress of service development. Accordingly, development teams should be capable of setting a release date for a service and then managing their development schedule to meet that date (more or less). Business sponsors may choose to slip a release date if the scope of the project changes or unforeseen risks occur that need to be dealt with before deploying the service. The development team's basic release capability at the managed level, gives business sponsors some visibility into service development and provides limited opportunities for course corrections and re-prioritization of requirements for subsequent development releases [22]. While the speed of service releases increases under a managed state of DevOps maturity, there is still enough latency to prevent release on demand. Projects that implement Agile development practices like regular integration and backlog management should find that they have the necessary process discipline to achieve a managed state, where even if they can't release new capability as fast as business sponsors need it, they can demonstrate regular process and confirm that an envisioned service will function as planned while meeting desired business objectives and performance measures [23].

At the **defined level of DevOps maturity**, the release process becomes a regular key indicator of project health. Most development teams that operate at this level of maturity create a build at least once a day from trunk, and developers will ensure they don't have more than a day's work sitting on a local branch in version control. Accordingly, branches to the main release trunk are short-lived, and the result is a significant reduction in system integration issues which tend to fester when individual developers or sub-teams maintain private builds or source code branches. At the defined level of maturity, deployment of new service versions is further accelerated by automating the process of provisioning integrated environments. In most cases, development teams at this level focus on automating deployment into a system test environment. When a team reaches the defined state of maturity, the result is a regular release cadence. The release time box is well defined, and it's easy to identify early warning sign that a project is in trouble. This makes it possible to exert early corrective actions like scope reduction, resource augmentation, and a schedule re-plan. While service delivery is much more predictable than the initial or managed level of maturity, it may still not be as fast as business leader's desire.

At the **measured level DevOps of maturity**, they reach a critical plateau where they can deploy a new release whenever one is needed. From a business perspective, the speed of service development meets or exceeds that capacity of the business to assimilate new services. At this level of maturity, software development team no longer hinders business innovation, but on the other side it enables it. As development teams reach this level of maturity, a shift in organization structure. Instead of vertically organized centres of excellence (e.g., business analysts, development, quality assurance/testing, infrastructure and operations/support), we see an organizational shift to cross-functional product teams. Individual roles become less important than the tasks that need to be completed to release new capability/service, and teams are likelier to self-organize their works. As part of this transition, the entire team assumes responsibility for service-level agreements associated with their product. And it's not just service testing and deployment that folds into the cross-functional organization. User experience should also be integrated into development teams at this level. At that level of maturity, teams find that they have eliminated all remaining bottlenecks in downstream build, test, and deployment phases. Deployments of individual changes in source code can now be performed in minutes, and there's no reason a team can't do multiple deployments to production in a single day. As software development teams move to this level of maturity, delivery teams prioritize keeping the main code trunk deployable over doing new work. Development teams don't let the integration build stay broken for more than a few minutes and anybody is empowered to revert breaking changes from version control. The focus of testing services also shifts up the development process. Test-driven development and acceptance-test-driven-development become core processes for development teams. These tests are layered on top of pre-flight builds to provide semantic validation of system function in addition to technical validation.

At the **optimizing level of DevOps maturity**, software development teams drive a continuous stream of incremental innovation. They form hypotheses about customer needs and how they can serve them, run experiments to test these hypotheses with customers, and then use feedback from their experiments to make design and service implementation decisions based on the best course of action. In this hypothesis-driven development model, software development teams

focus on optimizing cycle time in order to learn from customers in production, by gathering and measuring system feedback. Asynchronous services allow load balancing work distribution among different variants of the same service. Software development teams are able to inject probes into real-time production operations to monitor application load, deploying more resources as necessary at this level. Developers also test and evaluate the system as it functions in production — they may even initiate failures to make sure that the system takes proper corrective action. At the optimizing level, database changes are decoupled from application deployments, and individual service endpoints are decoupled from each other [29]. Software development teams at this level of DevOps maturity, can initiate business change by running experiments and proving business value. As the cost of each service deployment trends toward zero, the cost of running discrete experiments also drops significantly and becomes largely a factor of development labor costs.

III. PROPOSED DEVOPS MATURITY CALCULATOR ARCHITECTURE

The proposed calculator for the DevOps maturity is one of the innovative and quantitative tools that aim 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 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. To summarize the list of transformation capabilities, a brief description for each one along with the corresponding criteria will be listed as follows:

1. Operational management
    - Incident management methodology
    - Responsible teams
    - Troubleshooting and incident analysis approach
    - Incident monitoring and communication tracking
    - Incident notification and alerts mechanism
    - Incident response strategy.
2. Service management
    - Service management strategy
    - Adopting standards sharing approach
    - Collaboration cross teams
    - Communication style cross teams
    - Delivery model cross teams
3. Governance and process management
    - Service delivery model is available and up-to-date (including engagement model, org chart, team structure, R&R)
    - Service operation model is available and is reflecting the operation for the specific service
    - The governance model for service delivery is defined and standardized (including RACI model, accountability)
    - Standard SOW is available and is specific for the service
    - Standard SLA is available and is specific for the service
    - Estimation techniques are standardized
4. Build and continuous integration management
    - Build strategy

- Build and integration cross environments
- Versioning control for the build artifacts
- Reporting mechanism/hierarchy
- Build automation
- Automation approach for continuous deployments/integration
- Provisioning strategy
- Deployable status to mainline

5. Tools and automation

- Tooling matrix for build and run is defined and standardized
- Tools packages are identified for large, medium and small implementations (including tiers of tools, implementation plan, price & cost)
- Service automation and tooling support business objectives
- Tool matrix for automation is defined, and its implementation plan is established

6. Quality assurance and testing management

- Testing strategy
- Testing Automation and responsibility
- Testing phase involvement within development life cycle
- Development involvement through testing
- Release cycle impact with testing
- Regression bugs exists

7. Project and delivery management

- Project management strategy
- Delivery approach
- Prioritization for operational versus new features
- Collaboration support cross teams handling approach
- R&R (Roles and Responsibilities) is clear for all stakeholders

8. Collaboration and communication management

- Organization/process model
- Communication style
- Collaboration style between Dev/Ops
- Level of continuous improvement
- Team organization
- Process documentation
- Reporting hierarchy
- Metrics and measurement techniques

9. Feedback and continuous improvement

- Feedback loops strategy
- Monitoring and alerting applied on which components
- Service monitoring metrics applied level
- Alerting and monitoring scope
- Service failures proactive versus reactive

10. Vendor management

- Vendor management strategy
- Services management approach
- Handling of incident management

11. Continuous deployment management

- Deployment approach
- Fully automated
- Data based related deployment approach
- Release cycle and cadence
- Deployment cross environments mechanism

12. Configuration management

- Configuration management strategy
- Versioning control for the environments
- Configuration Items control and management

13. Technology and architecture management

- Technology style
- Custom versus rigid based
- Static versus dynamic

14. Change management

- Change management strategy
- Change management control board formulation
- Change automation
- Review process for each change
- Feedback loops implemented
- Data migration strategy

IV. DEVOPS MATURITY CALCULATOR USE CASE

To show the value of the proposed DOMC tool, a use case is used to show how the tool is working via real project which assessed against the 14 DevOps maturity capabilities. For illustrative purpose within the paper, I'll show only one capability and the same workflow can be applied similarly against the rest of capabilities. The DOMC tool is design with nice friendly graphical interface (GUI) and hosted on cloud server to facilitate easily accessibility from wherever. DOMC is developed using PHP/Java programming languages along with My SQL DB to support saving the organization/project data results.

Step 1:

User needs to login to the cloud server using his/her credential to select one of two options, either to create a new template for his/her project or to use an existing template as shown in Fig 2.

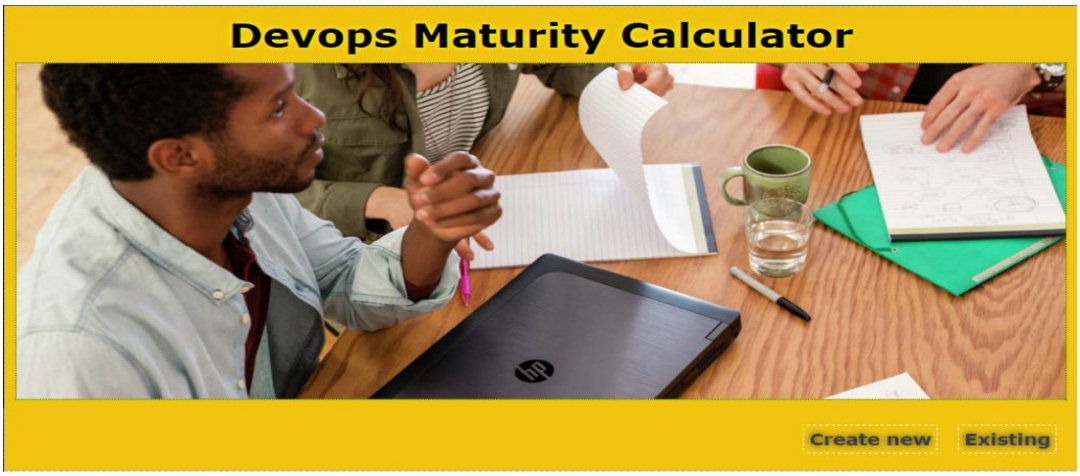

FIG 2. DOMC LOGIN INTERFACE

Step 2:

For the purpose of this use case, we selected to start from scratch where no existing project and start to create new project. For this purpose user needs to select the type of template and enters some basic information related to the project like project name, project description, domain, project manager name, and any relevant comments/remarks as shown in Fig 3.

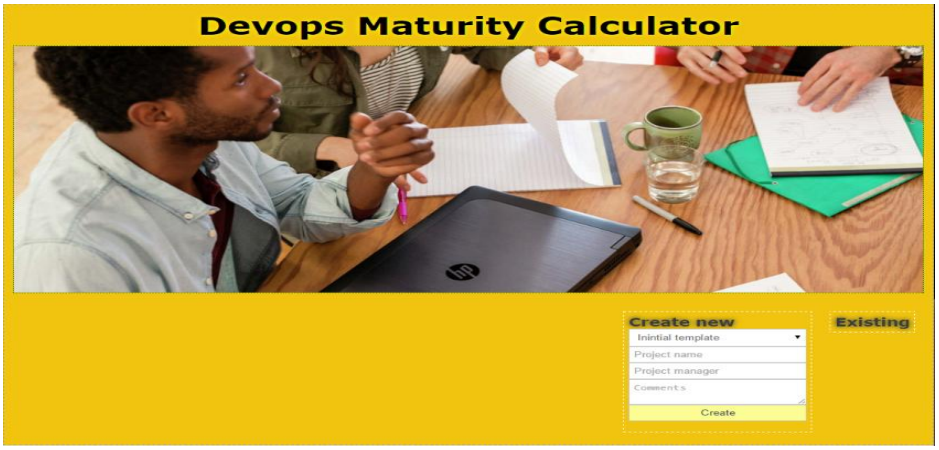

FIG 3. DOMC LOGIN INTERFACE WITH PROJECT DETAILS

Step 3:

1- Once user enters his/her project details, the DOMC tool will create a new project template with all the 14 capabilities with the corresponding criteria such that to enter the project current behaviour against one of the criteria. In the below case as shown in Fig 4, I selected the operational management capability with different criteria like (Incident response strategy, Incident management methodology, Responsible teams, Troubleshooting and incident analysis approach, Incident monitoring and communication tracking, Incident notification and alerts mechanism).

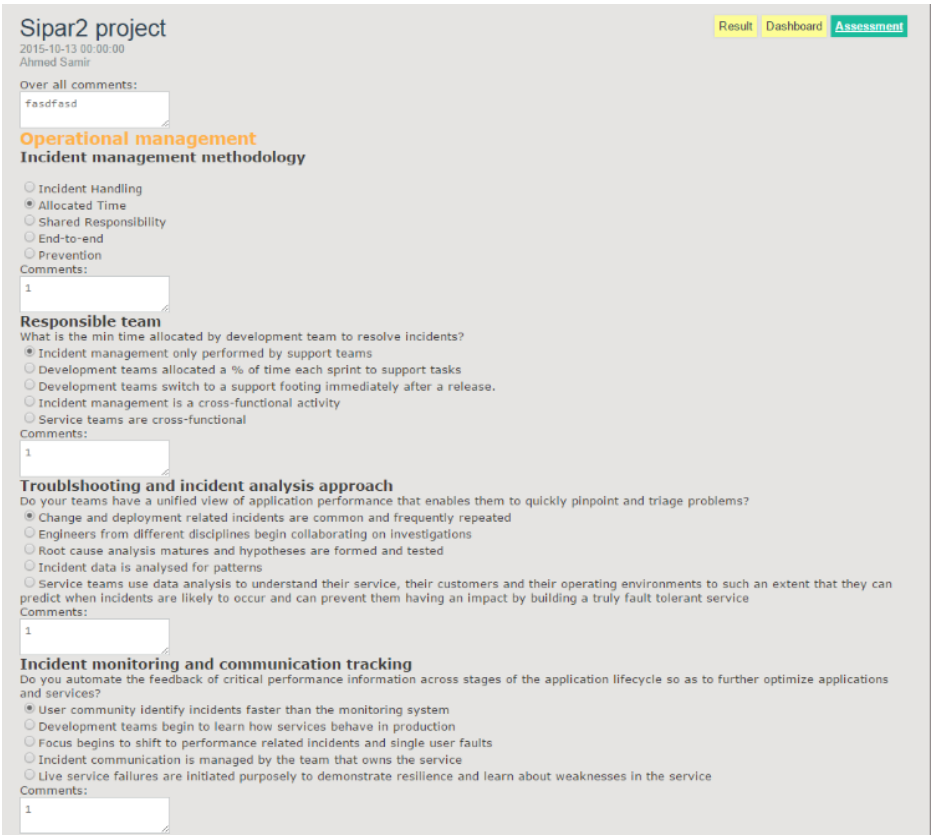

FIG 4. DOMC CAPABILITY AND DETAILED CRITERIA INTERFACE

2- User then needs to select the best match between his/her project behaviour 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. For example in the below example the incident management methodology is mapped in allocated time which is level 2 of maturity or (managed level) as described in section 2, while responsible team is mapped into (incident management is only performed by the responsible team) which is level 1 of maturity or (initial level).

3- Besides the score per criteria like what mentioned above, the tool also calculate a score on the capability level for example for the operational management capability consolidating the score calculated for each and every criteria which gives the average score per all criteria within the corresponding capability. For example in the below case the average score calculated to be around 25% because most of the criteria are on the (initial level).

Step 4:

1- User will then needs to repeat the process mentioned in step 3 for all other capabilities and their corresponding criteria.

2- DOMC will then calculated overall score for the current project behaviour mapped against DevOps standard represented with the 14 capabilities as mentioned above.

3- DOMC will then mapped 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.

4- In our use case, we based the calculations on the following settings in Table 1, where final overall score compared against below ranges to calculate the maturity level.

TABLE 1
MATURITY LEVEL THRESHOLDS SETS

Maturity Level	Threshold
L1	X <= 40%
L2	40 %< X <=65%
L3	65 %< X <=80%
L4	80 %< X <=92%
L5	X >= 92%

Step 5:

DOMC produces different statistics to evaluate the project/organization maturity against the different capabilities. As shown in first pie chart 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.

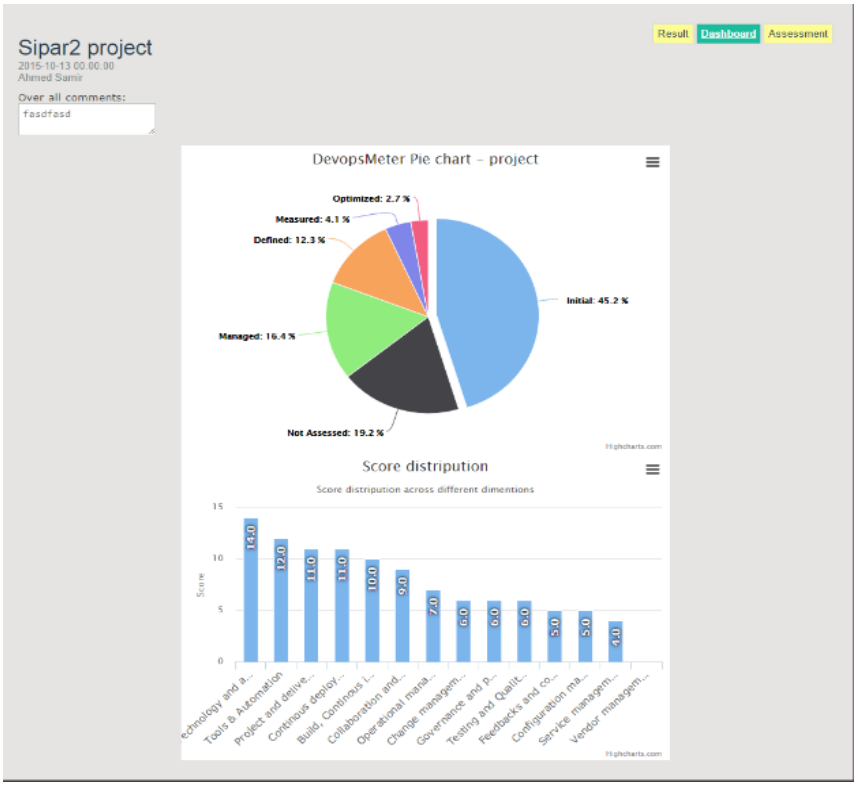

FIG 5. DOMC STATISTICS RESULTS

The second levelling chart shows in descending order the score corresponding to each capability from the overall 14 capabilities listed as part of the DevOps maturity model. This score is based on different factors like each capability criteria score, weighting assigned to each criteria that shows how important/priority this criteria compared to others within the same capability. This type of graph is vital to enable the organization detecting the capability where it suffer from and still has some gaps need to be tackled besides highlighting those where the project has strengths to build upon throughout the DevOps transformation journey for the organization.

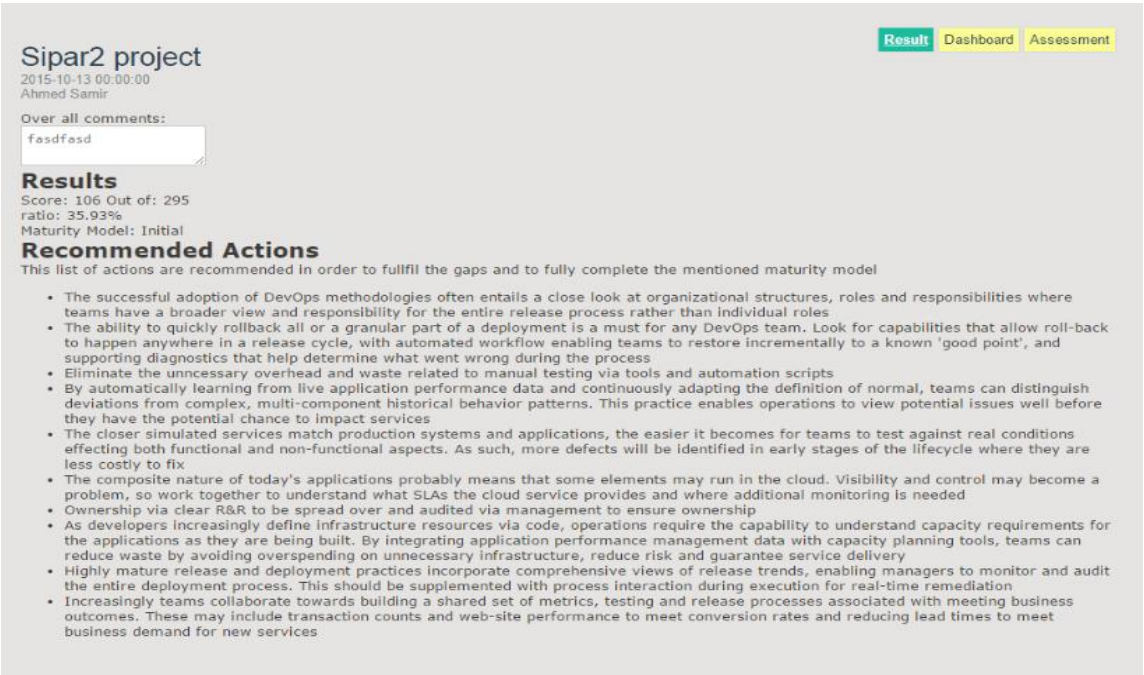

FIG 6. DOMC OUTCOMES AND SCORING SCHEME

Step 6:

DOMC provides more features to help the organization to start working in the execution phase of the transformation objective by drafting set of recommended actions for those gaps where the project has weaknesses based on the scoring scheme of each and every capability criteria as shown in the above statistics graphs shown in Fig 5. Based on our use case there are many recommended actions in different domains because the project maturity level still in initial level with overall score of 35% or 106 out of 295 as shows in Fig 6. Adopting these recommended actions will help the organization close current gaps and move to the next maturity level overall especially for those capabilities with overall score less than next level threshold as indicated in table 1.

There is one other feature provided by DOMC helping the organization best detect the gaps of each criteria via KPIs (Key Performance Indicators) and build the plan to close those gaps via CSFs (Critical Success Factors) based on metric based approach [25]. Through this the project manager will measure the KPIs correspond to each criteria and compare against those given by DOMC to assess how much it covers in the DevOps scale for the corresponding criteria. And based on the outcomes from this criteria assessment, the project manager uses the CSFs to build action plan to close any existing gaps.

This process can then be repeated regularly every six months or so based on the maturity of the organization and project budget to apply such transformations while satisfying the delivery commitments towards business users and clients.

V. CONCLUSION

The uniqueness of the proposed tool or DOMC (DevOps Maturity Calculator) is three folds, first the simplicity of the proposed mechanism to assess and transform based on the goals set by the organization/project in quantifiable or metric-based approach. Second, adopting quantitative approach to measure the maturity and progress of the project for each and every capability. Third, Using CSFs, KPIs, and set of recommended actions to guide the project towards higher maturity level in smooth and seamless manner. Through DOMC organizations will be able to imbed the DevOps maturity model into their business process to satisfy their clients demand through both measuring and assessing the DevOps maturity level using set of capabilities/criteria. The objective of this work is basically target to standardize the DevOps process/best practices to be adopted by most of current IT organization due to the main value/need expected from DevOps to fulfil current market demand. Future plan based on work introduced in this paper will be focused to improve the DOMC tool and add more features like linking with other DevOps tools that can feed the data for building the DOMC metrics in an automated manner. Besides building historical data base for each project to track the progress over different assessment and provide the statistics graph to show how project/organization evolve over specific period of time.

REFERENCES

[1] L. Fitzpatrick and M. Dillon. The business case for devops: A five-year retrospective. Cutter IT Journal, 24(8):19{27, 2011.
[2] 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.
[3] J. Humble and J. Molesky. Why enterprises must adopt devops to enable continuous delivery. Cutter IT Journal, 24(8):6{12, 2011.
[4] B. Keyworth. Where is it operations within devops? Cutter IT Journal, 24(12):12{17, 2011.
[5] B. Kitchenham. Procedures for performing systematic reviews, 2004.
[6] A. Le-Quoc. Metrics-driven devops. Cutter IT Journal, 24(12):24{29, 2011.
[7] M. Loukides. What is DevOps? O'Reilly Media, Sebastopol, CA, 2012.
[8] 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.
[9] B. Phifer. Next-generation process integration: CMMI and ITIL do devops. Cutter IT Journal, 24(8):28{33, 2011.
[10] H. Pruijt. Multiple personalities: the case of business process reengineering. Journal of Organizational Change Management, 11(3):260{268, Jan. 1998.
[11] J. Roche. Adopting devops practices in quality assurance. Communications of the ACM, 56(11):38{43, 2013.
[12] A. Schaefer, M. Reichenbach, and D. Fey. Continuous integration and automation for devops. Lecture Notes in Electrical Engineering, 170 LNEE:345{358, 2013.
[13] O. Akerele, M. Ramachandran, and M. Dixon. System dynamics modeling of agile continuous delivery process. In Proceedings - AGILE 2013, pages 60{63, 2013.
[14] 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.
[15] 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.
[16] D. Cukier. Devops patterns to scale web applications using cloud services. In Proceedings - SPLASH '13, pages 143{152, Indianapolis, Indiana, USA, 2013.

[17] P. Debois. Opening statement. Cutter IT Journal, 24(8):3{5, 2011.
[18] D. DeGrandis. Devops: So you say you want a revolution? Cutter IT Journal, 24(8):34{39, 2011.
[19] D. Feitelson, E. Frachtenberg, and K. Beck. Development and deployment at facebook. IEEE Internet Computing, 17(4):8{17, 2013.
[20] W. Shang. Bridging the divide between software developers and operators using logs. In Proceedings - International Conference on Software Engineering, pages 1583{1586, 2012.
[21] 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.
[22] SACKS, M. DevOps principles for successful web sites. In Pro Website Development and Operations. Springer, 2012.
[23] Samer I. Mohamed DevOps shifting software engineering strategy - value based perspective, manuscript *Volume 17, Issue 2, Ver. IV (Mar – Apr. 2015), PP 51-57.*
[24] *Heiko Koziolek. Goa, Question Metric, solume 4909 of the series lecture notes in computer science pp 39-42.*
[25] Samer I. Mohamed Goal oriented DevOps transformation framework – Metric phased approach, manuscript *Volume 17, Issue 2, Ver. IV (Mar – Apr. 2015), PP 51-57.*
[26] 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.
[27] M. Walls. Building a DevOps Culture. O'Reilly Media, Sebastopol, CA, 2013.
[28] 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.
[29] 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