A Research on DevOps Maturity Models

Mohammad Zarour, Norah Alhammad, Mamdouh Alenezi, Khalid Alsarayrah

Abstract*: Despite the big number of software process models currently available which have been used and practiced for many years, we could not till now totally solve the problem of projects' late submissions! Meanwhile Software have constantly become bigger, more complex, and require high quality. A recently developed model, called DevOps, aims at producing fast and high-quality releases by bringing the development and operation team to work together. Unfortunately, DevOps is still lacking a clear definition as well as empirical studies that document the experience in implementing and enhancing it. Maturity models are used as a tool to assess the effectiveness of an organizational processes on adopting certain practices and identify what capabilities they need to acquire next in order to improve their performance and reach higher maturity level. However, there are few DevOps maturity models which have been emerged as means to assess DevOps adopted practices. This research aims to identify and benchmark the DevOps maturity models available in literature. We were able to identify several maturity models and compare among them.

***Index Terms*: DevOps, Comparison, Maturity Model, Process Model.**

I. INTRODUCTION

Despite the big number of software process models currently available which have been used and practiced for many years, we still could not totally solve the problem of projects' late submissions! Meanwhile Software have constantly become bigger, more complex, and require high quality. A recently developed model, called DevOps, aims at producing fast delivery to customers by bringing the development and operation team to work together.

DevOps is the new software process that extends the agility practices within the collaborative culture to enhance the process of software development and delivery. DevOps is concerned with improving the collaboration between the development and operation teams to achieve fast high-quality releases. Although DevOps is in use now for several years, but it is still in its infancy period [1] where this new philosophy to develop software is still lacking a clear definition [2], [3] as well as empirical studies that document the experience of its implementation worldwide.

Despite the increasing adoption of DevOps where organizations have different motivations to adopt it, DevOps requires further investigation in assisting the quality of the adoption, as there is few DevOps maturity model that gauges the maturity. Recently, few DevOps maturity models have been emerged as means to assess DevOps adopted practices. However, the reported experience on using these maturity models for DevOps in literature is scarce. Same thing applies for the assessment methods for these DevOps maturity models where the literature lacks detailed description of these methods that prescribe how to assess the DevOps adoption for organizations to improve their maturity incrementally.

This research will study the available maturity models of DevOps that are documented in the literature, compare between them to identify the strengths and weaknesses of each one, check the assessment methods available based on these maturity models. Such study is crucial for process assessors to know the various maturity models available and decide which one to be uses for the proposed process assessment initiative.

II. DEVOPS MATURITY MODELS

Since the nineties of the previous century, software organizations have shown growing interest in assessing and improving their software process using various well-established maturity models that includes CMM, CMMI and ISO/IEC 15504 [4]. It is observed that the maturity assessment is an expensive and arduous activity for organizations and more work is needed to automate this process [5]. While researchers and practitioners are working to better understand the software processes, their best practices and ways to assessing them, new software process models emerge with their own practices. This would increase the burden on software process engineers to define and practice the maturity models for the new emerged process models.

Note that this paper's focus is not to conduct a systematic literature review as the DevOps concept is new and the maturity models related to it are few. Hence neither the systematic literature review nor the mapping studies are used in this research. A simple search in the main databases that includes IEEE, ACM, Springer, and google scalar is enough and serve our purposes. Unpublished work or thesis work is excluded.

We were able to identify seven maturity models, namely: [6]–[12]. Note that four out of the seven identified maturity models are documented as white papers which raises a threat to this study hence their validity and applicability are questionable. At the same time, we believe that this is a strong driver for more theoretical and empirical research related to

Revised Manuscript Received on September 19, 2019.

Mohammad Zarour, College of Computer and Information Sciences, Prince Sultan University , Rafha Street, Riyadh, Saudi Arabia, E-Mail: mzarour@psu.edu.sa

Norah Alhammad, College of Computer and Information Sciences, Prince Sultan University , Rafha Street, Riyadh, Saudi Arabia, E-Mail: norah.a.alhammad@psu.edu.sa

Mamdouh Alenezi, College of Computer and Information Sciences, Prince Sultan University , Rafha Street, Riyadh, Saudi Arabia, E-Mail: malenzi@psu.edu.sa

Khalid Alsarayrah, College of Computer and Information Sciences, Prince Sultan University , Rafha Street, Riyadh, Saudi Arabia, E-Mail: khalidtorki@psu.edu.sa

DevOps concepts and adoption that should follow this research work.

2.1 IBM DevOps Maturity Model

Bahrs, P form IBM [6], provided a thorough analysis on the adoption of IBM DevOps approach for promoting continuous delivery of software. The author identified four dimensions in adopting or implementing continuous software growth within an organization. These dimensions include Planning and measuring, Developing and testing, Releasing and deploying, and Monitoring and optimizing. The IBM DevOps maturity model is a practice-based and reflects a wider context within the adoption framework of an organization. It focuses on defining the best practices to be applied in the adoption of new software solutions iteratively.

A well-articulated approach for assessing current DevOps practices within an organization is also provided in [6]. It also helps in defining a clear roadmap for DevOps implementation. Furthermore, the mentioned research work provided its readers with a high-quality approach for measuring the improvement made by an organization in implementing the IBM DevOps approach. Most importantly, this DevOps maturity provides a clear set of steps for preparing, piloting and releasing system improvements within an organization.

It is important to note that his model does not specify the applicability of the IBM DevOps approach in other software platforms that do not run on IBM software. Another limitation is that it does not provide a clear justification on the investment strategy for achieving DevOps maturity. IBM DevOps maturity model has 4 levels, as follows, See Fig. 1:

Level-1 is "Practiced": At this level, the enterprise standards are not defined, inconsistent automation, and teams may perform some activities associated with the practice inconsistently.

Level-2 is "Consistent": The enterprise standards at this level are defined, automation follows the standards, and teams perform activities associated with the practice according to the standards.

Level-3 is "Reliable": At this level, enterprise's standards are being followed, an exist mechanisms to assist adoption, a mentor team is available to assist in adopting the best practices.

Level-4 is "Scaled": At this level, institutionalized practices are defined for the adoption across the enterprise, matured core team is formulated, and feedback process is established for the standards improvement.

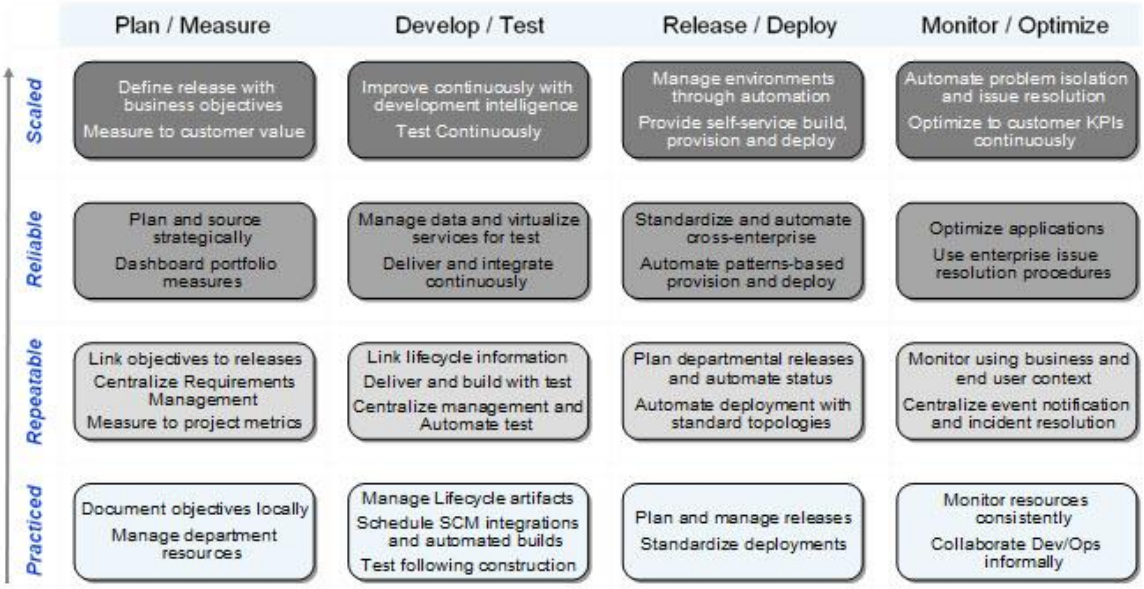

Fig.1. IBM DevOps Maturity Model [6]

2.2 MOHAMED DEVOPS MATURITY MODEL

Mohamed, S [12], has introduced a new DevOps maturity model and then assessed how the model can affect the existing global software engineering practices and processes. The proposed DevOps maturity model is based on the Capability Maturity Model Integration (CMMI) and it is composed of five maturity levels against four dimensions that include quality, automation, collaboration and governance.

Mohamed, S clarified that the implementation of the CMMI based DevOps maturity model helps in improving operational efficiency, increase visibility and mitigating significant risks such as downtime during implementation. The strength of this model is that, using techniques of the

CMMI model helps in identifying the capability of the DevOps model at each level of its maturity. It also provides a clear transformation framework as the DevOps model matures from one phase to another. The maturity model is defined as follows, See Fig. 2:

Level-1 is "Initial": At this level, ad-hoc communication with no clear process, no automation implemented, uncontrolled governance/process where the outcome of any service is not predictable, and no quality standards exist. The whole activities are done based on the process owner objectives.

Level-2 is "Managed": At this level, communications are controlled but not shared between teams, documented automation process but not executed, executed governance but not standardized, and ad-hoc quality management is in place.

Level-3 is "Defined": At this level, the communication, automation, and governance are standardized. Quality standard exists.

Level-4 is "Measured": At this level, the communication metrics, automation metrics, and governance metrics, quality metrics exist for improvement and measurement.

Level-5 is "Optimized": At this level, constructive communication environment, tools and processes are adopted, smart automation to maximize throughput, optimized governance self-adaption, and continuous quality improvement.

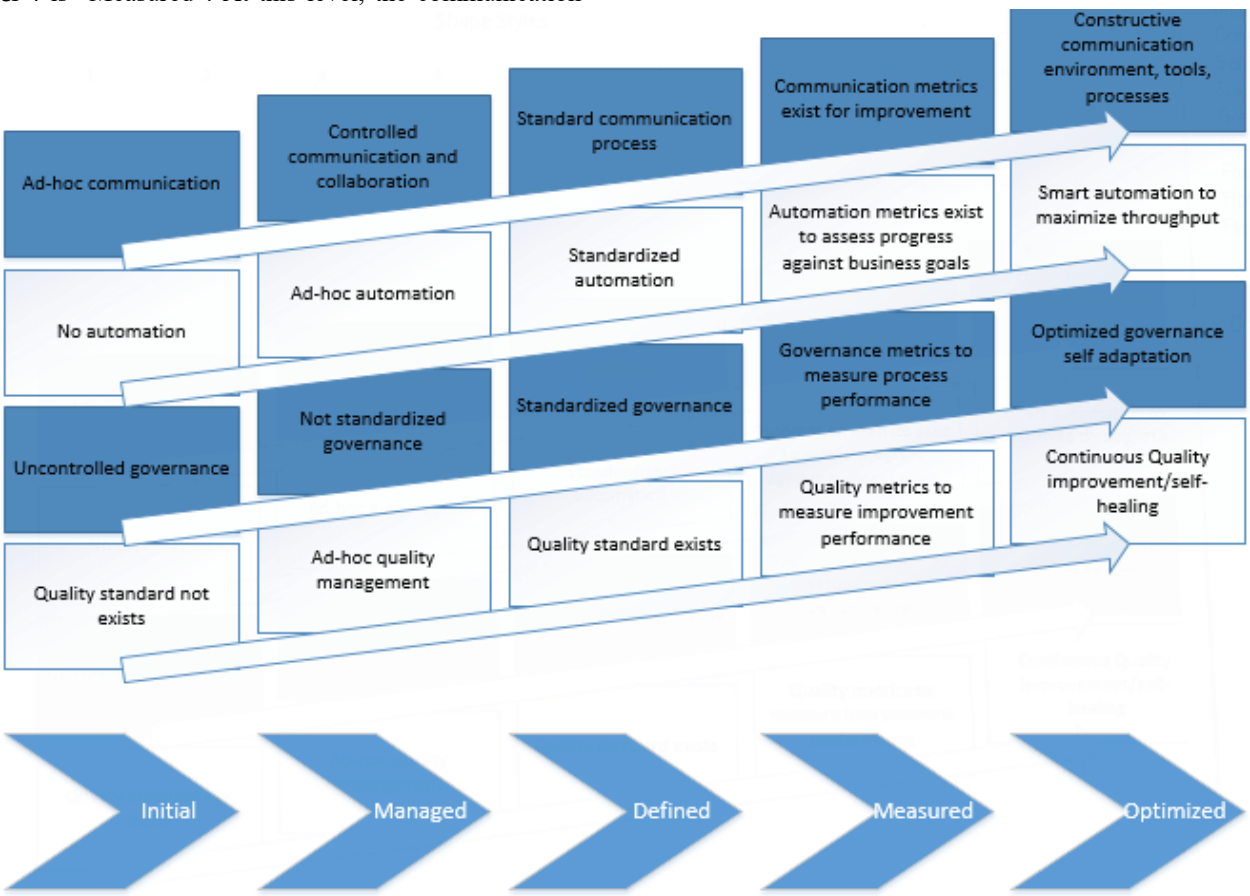

Fig. 2. Mohammed's DevOps Maturity Model [12]

2.3 CAPGEMINI DEVOPS MATURITY MODEL

G. Menzel and A. Macaulay [11] from Capgemini, designed DevOps maturity model that enables businesses to identify the current maturity level. This model has five levels of maturity that measure three dimensions which are people, process and tools. It is defined as follows, See Fig. 3:

Level-1 is "Basic": at this level, the strategy, design, development, and testing are separated. Teams focus on their goals and objectives, ad-hoc process, all activities are manual, no automation tools, and no integration and sharing.

Level-2 is "Emerging": teams are separate, developers focus on functional and less focus on the non-functional requirements, establish a managed process that is restricted to a specific environment, and automatic scripts are developed for some environments such a development environment.

Level-3 is "Co-ordinated": here, operational team are engaged in the first phases. There are joint processes for the development and operational aspects. The environment setup and characteristics are partially understood and automated

Level-4 is "Enhanced": The entire solution lifecycle from design, build, test to run has been covered by the joint team. There is a single process for the entire solution lifecycle. The environment setup and characteristics are clear and most of the setups for development, testing and operation are automated.

Level-5 is “Top level”: One collaborative team with full knowledge sharing. There is a single process that cover the entire solution lifecycle and organization strategy. Setup for all environments are automated from one single repository.

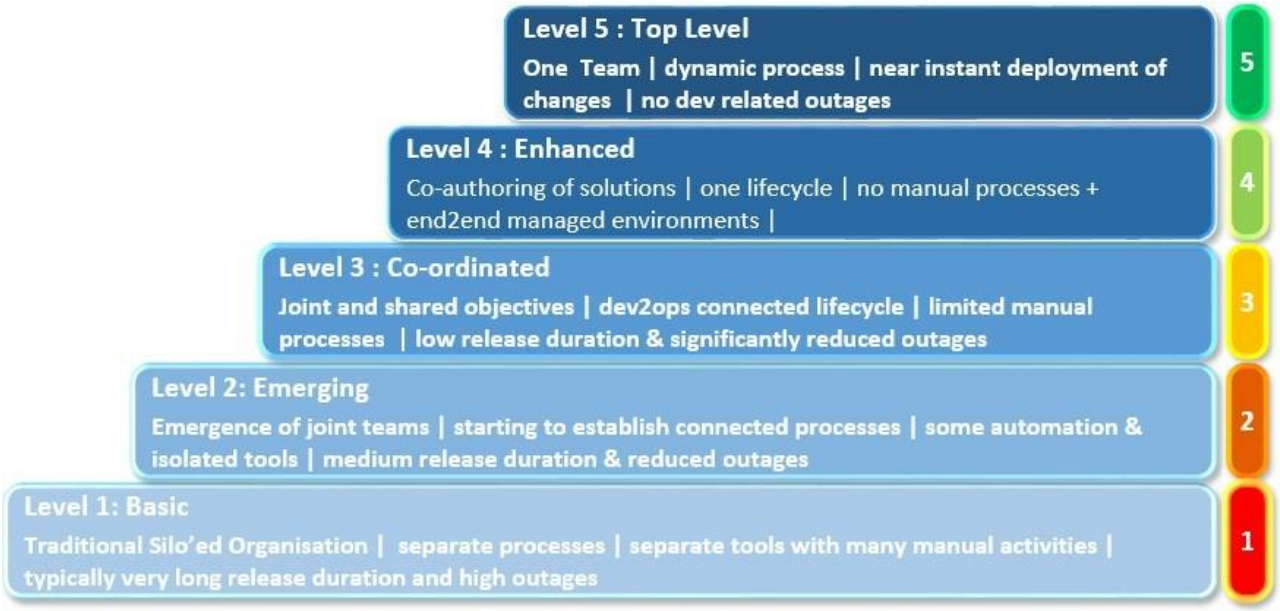

Fig. 3. Capgemini’s DevOps Maturity Model [11]

2.4 Hewlett Packard Enterprise DevOps Maturity Model

Inbar et al. [10] from Hewlett Packard Enterprise (HPE), developed a new maturity model that is aligned with the CMMI maturity model to measure DevOps adoption. This model is designed to cover the entire lifecycle of an application for large organizations. It is applied to measure the process, automation, and collaboration dimensions. The maturity model is defined as follows, see Fig. 4:

Level-1 is “Initial”: The collaboration is poor, ad-hoc team communication, and independent stakeholders’ decisions, no automation processes, and unrepeatable processes.

Level-2 is “Managed”: The collaboration is managed, communication and coordination are managed, the process is partially automated and documented and is not standardized across projects

Level-3 is “Defined”: The collaboration is established between the teams, central automated infrastructure, automation is tailored for application and environments. processes are characterized and standardized across projects.

Level-4 is “Measured”: The collaboration is measured based on processes communication to identify bottlenecks, the process is automated, measured, and controlled. The process is visible and predictable of entire process.

Level-5 is “Optimized”: The collaboration is optimized and effective knowledge sharing and individual empowerment. Continuous improvement for the automated process, continuous assessment for the entire process, and minimize risk and cost for the business objectives.

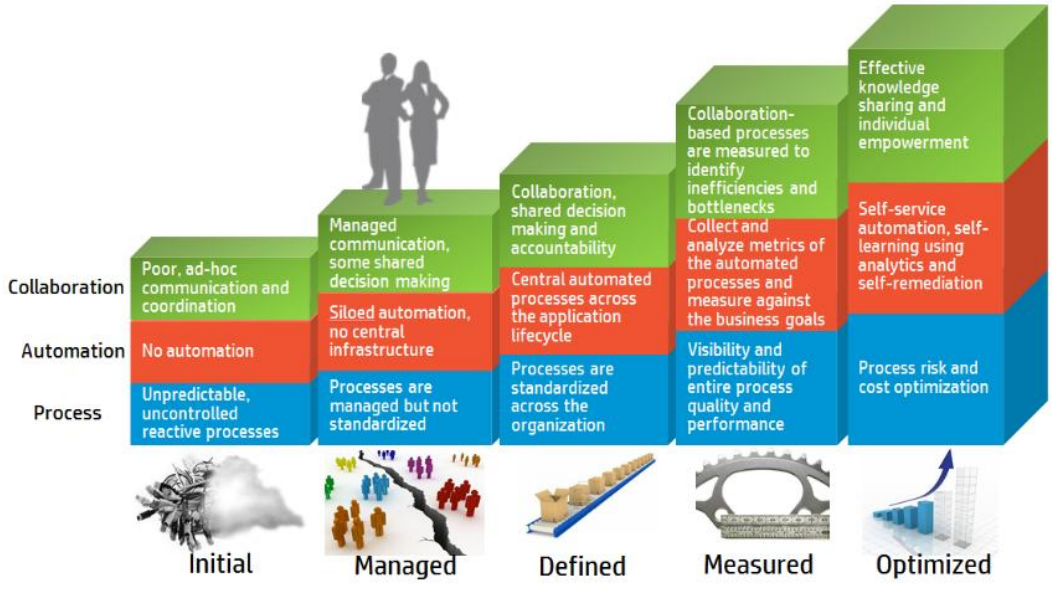

Fig.4. Hewlett Packard Enterprise’s DevOps Maturity Model [10]

2.5 Bucena DevOps Maturity Model

Bucena and Kirikova [12], have developed a DevOps maturity model based on CMM approach, which consist of five maturity levels. Each of these levels has four dimensions which are technology, process, people and culture. It is defined as follows, See Fig. 5:

Level-1 is "Initial": The environments, tests, data migration, and deployment are performed manually. The delivery process and project management are inconsistent, ad-hoc approaches for learning. Communication is restricted, and lack of awareness as how the culture impacts day to day business.

Level-2 is "Repeatable": The environments configurations are externalized and versioned, the delivery process is scheduled, project and requirement are managed, requirement is based on testing, development documents are up-to-date, scrum development, and managed processes but not standardized, the team organized around deliveries, In the culture dimension, the communication among internal team are rapid.

Level-3 is "Defined": Environments virtualization is adopted. The delivery process is automated, and integrated. The team is organized around projects, the communication between teams are rapid, clear project requirements, active collaboration, and identified culture traits.

Level-4 is "Managed": The environments are managed effectively, smoked tests shared with operation team, production deployment is automated. The process delivers frequently, visible and predictable. The team organized around products, frequent collaboration and communication, clear product requirements, and culture viewed as asset to be managed.

Level-5 is "Optimized":

The environments fully automated, continuous work on process improvement for better visibility and faster feedback, continuous delivery process, and the collaboration between operation and development teams to manage risks and reduce cycle time. Teams are organized around KPIs. Fig.5 illustrates a sample of this maturity model.

Area	ID	Initial level (1)	Repeatable level (2)	Defined level (3)	Managed level (4)	Optimized level (5)
TECHNOLOGY	T1	Environments are provisioned manually	All environment configurat-ions are externalized and versioned	Virtualization used if applicable	All environ-ments managed effectively	Environment provisioning fully automated
	T2	Manual tests or minimal automation	Functional test automation	Triggered automated tests	Smoked tests and dashboard shared with Op.t.	Chaos Monkey
	T3	Data migration un-versioned and performed manually	Changes to DB done with automated scripts versioned with application	DB changes performed automatically as part of deployment process	DB upgrades and rollbacks tested with every deployment	Feedback from DB performance after each release

Fig. 5. Sample of Bucena's DevOps Maturity Model [7]

Eficode Maturity Model

Eficode's maturity model [9], see Fig. 6, has five dimensions which are: organization and culture, environments and release, builds and continuous integration, quality assurance, and visibility and reporting. The model defines four maturity levels. At the first level, DevOps practices are not used. At the second and third level, organization has started to implement some DevOps practices, but they are in their early stages and there are room for improvement. The last level in Eficode's model focuses on the use of metrics for continuous improvement to achieve efficiency and quality where the DevOps practices become in an ideal state.

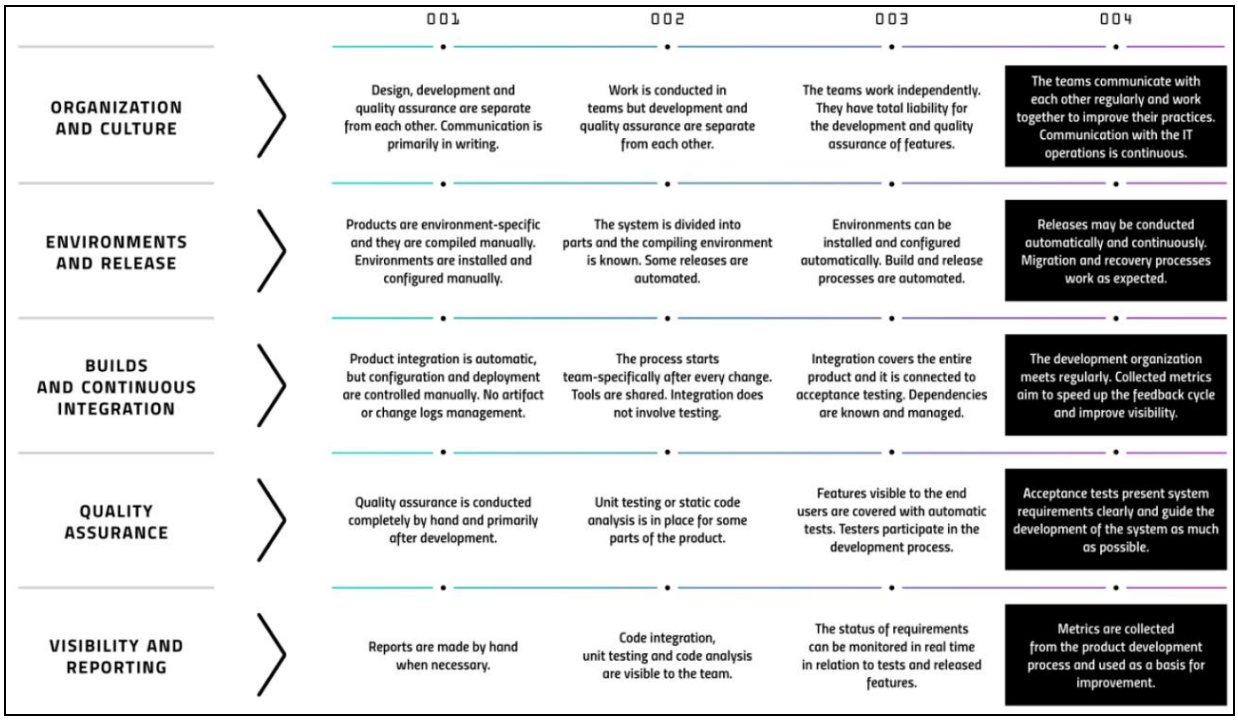

Fig.6. Eficode DevOps Maturity Model [9]

2.7 Feijter Maturity Model

Feijter DevOps maturity model [8], shown in Table-1, includes focus areas that enables software production organizations to mature in a fine grain manner. The model is dedicated to be used by software product organizations (SPO) that produces software to be used by several customers but not a customized software for specific customer. Feijter model includes sixty-three capabilities (represented as letters in the model) and are distributed over ten capability levels. A case study was carried out, at Centric organization, to experience the maturity model in practice. The model consists of three main dimensions namely: culture and collaboration, product and process quality, and foundation.

III. COMPARISON AND DISCUSSION OF THE DEVOPS MATURITY MODELS & RESULTS

The comparison among DevOps maturity models is based on three factors which are:

Maturity models' levels names.

Maturity models' number of levels, publication year, number of dimensions, and application

Maturity models' dimensions

The result of this comparison and the following discussion are informative to identify the strengths and weaknesses of the existing maturity models and to decide which maturity model to use in any DevOps process assessment activity.

Table 1. Feijter et al. DevOps Maturity Model [8]

Focus area \ Level	0	1	2	3	4	5	6	7	8	9	10
Culture and collaboration											
Communication		A				B	C			D	E
Knowledge sharing				A		B	C				D
Trust and respect							A	B	C		
Team organization		A	B						C	D	
Release alignment				A					B	C	
Product, Process and Quality											
Release heartbeat		A				B	C		D	E	F
Branch and merge			A	B		C		D			
Build automation			A	B		C					
Development quality improvement			A			B		C	D	E	
Test automation				A	B	C			D		E
Deployment automation					A	B		C			D
Release for production					A			B	C	D	
Incident handling			A					B	C	D	
Foundation											
Configuration management			A	B		C					
Architecture alignment			A					B			
Infrastructure				A			B	C	D		

3.1 Maturity Models' Levels Names

From our investigation of the DevOps maturity models, we have noticed that, see Table 2:

There are two similar models which are: Mohamed's model, and Inbar's model because both of them comply with CMMI.

There are three similarities in the levels' names between Mohamed's model, Inbar's model, and Bucena's model which are level 1,3, and 5.

The level names are different from one maturity model to another but their meanings, may have some similarities and differences. For instance, the first level (initialization) for Bahrs's model is named "Practiced" while in Menzel's model it is named "Basic", and the others name it "Initial".

Eficode and Feijter models used numbers to name the maturity levels

The fourth level has different names, e.g. scaled, managed, enhanced or measured, but they all reflect the same meaning of having a managed process:

Level 1: A level for the starting point for use of the new process (Initial).

Level 2: A level where the process is at least documented sufficiently (Managed).

Level 3: A level where the process is defined as a standard process (Defined).

Level 4: A level where the process is managed in accordance metrics (Measured).

Level 5: A level where the process managed efficiently (Optimized)

Table 2. Maturity Models' Levels

Maturity model	Level 1	Level 2	Level 3	Level 4	Level 5
Bahrs's model	Practiced	Consistent	Reliable	Scaled	NA
Mohamed'smodel	Initial	Managed	Defined	Measured	Optimized
Menzel's model	Basic	Emerging	Co-ordinated	Enhanced	Top level
Inbar model	Initial	Managed	Defined	Measured	Optimized
Bucena model	Initial	Repeatable	Defined	Managed	Optimized
Eficode's model	1	2	3	4	NA
Feijter model	Differ than other models with 10 maturity levels				

3.2 Maturity Models' Number of Levels, Publication Year, Number of Dimensions, and Application

From Table 3, we noticed that the maturity models have different number of levels:

Two maturity models have four levels, Bahrs's and Eficode's models.

Four maturity models have five levels, which are: Mohamed's model, Inbar's model, Menzel's model and Bucena's model.

One odd model has 10 maturity models that is Feijter model.

Regarding the maturity models' publishing year, note that all the models are relatively new and published in the period of 2013-2018 which justifies the scarce of published work related to DevOps, which we expect to increase the coming few years. Furthermore, regarding the number of dimensions, note that all models have 3-5 possible dimensions, as discussed in the next section.

Table 3. Maturity Models' Number of levels, Year

Maturity model	Number of levels	Pub. Year	Number of dimensions	Organizations Validated the model
Bahrs's model	4	2013	4	1 (IBM)
Mohamed's model	5	2015	4	-
Menzel's model	5	2015	3	-
Inbar's model	5	2013	3	1 (Hewlett Packard)
Bucena's model	5	2017	4	1 Anonymous SMEs
Eficode's model	4	2015	5	
Feijter Model	10	2018	3	1 (Centric)

3.3 Maturity Models' Dimensions

Different DevOps maturity models have different dimensions, see Table 4:

Four maturity models have Process dimension which are: Mohamed's model, Menzel's model, Inbar's model, and Bucena's model.

In the second dimension (dimension B), Mohamed, S [3], and Inbar et al. [5] models have automation dimension, where G. Menzel and A. Macaulay [4] has tool dimension, Bucena [6] has a technology dimension and Eficode [8] has a environment and release dimension that contains automation and tools as sub dimension.

Most of the maturity models have dimension for team collaboration, as in Mohamed, S [3] and Inbar et al. [5]. G. Menzel and A. Macaulay [4], Bucena [6] call it people and differentiate it from the culture dimension, Eficode and Feijter models call it culture dimension.

Bahrs, P [2] model dimensions revolve on the process automation

One model has fifth dimension related to visibility and reporting

Table 4 Maturity Models' Dimensions

Maturity model	Dimension A	Dimension B	Dimension C	Dimension D	Dimension E
Bahrs's model	Plan	Develop	Release	Monitor	-
Mohamed's model	Collaboration	Automation	Process / Governance	Quality	-
Menzel's model	People	Tool	Process	-	-
Inbar's model	Collaboration	Automation	Process	-	-
Bucena's model	People	Technology	Process	Culture	-
Eficode's model	Organization and culture	Environments and release	Builds and continuous integration	Quality assurance	Visibility & reporting
Feijter Model	Culture & Collaboration	Product, Process & Quality	Foundation	-	-

IV. CONCLUSION AND FUTURE WORK

To conclude, it is clear from the conducted comparison that most of the maturity models have 5 levels either following CMM or CMMI, others have 4 and one exceptional model has 10. Regarding the experiment application two of the models are applied and validated by the institute that already developed the mode while other to models are validated by applying the model in one independent organization. This means that the usage of the DevOps maturity models is still very limited and is not yet used by variety of organizations and this would make the validity of such models questionable. Accordingly, more research and empirical work is vitally needed to practice and validate the proposed DevOps maturity models.

It is also noted that there are large similarities in the measured dimensions in most of models expect Bucena's and Feijter's models. Both models assist the culture dimension and both of them have validated their model in one organization from the industry, i.e. an independent organization that they do not work for. Moreover, we found that Bucena's model is holistic and covers all of DevOps dimensions while Feijter's model is dedicated for SPO organizations. Hence, we believe that Bucena's and Eficode's models are comprehensive and promising models to build on them and conduct future assessment to assess DevOps maturity.

Another observation concerning the application of the various DevOps maturity models, is that none of the researchers have documented the adopted assessment method in an academic publication. All of them documented the model and its applications then discussed the findings and results. We believe that this is not enough. Developed assessment method should also be published to be used or enhanced by other researchers. The next step is to use one of the recommended models to assess the maturity of Saudi organization in adopting DevOps via an empirical study.

V. REFERENCES

1. P. Rodríguez et al., "Continuous deployment of software intensive products and services: A systematic mapping study," J. Syst. Softw., vol. 123, pp. 263–291, Jan. 2017.
2. F. M. A. Erich, C. Amrit, and M. Daneva, "A qualitative study of DevOps usage in practice," J. Softw. Evol. Process, vol. 29, no. 6, p. e1885, Jun. 2017.
3. J. Sharp and J. Babb, "Is Information Systems Late to the Party? The Current State of DevOps Research in the Association for Information Systems eLibrary," in AMCIS 2018 Proceedings, 2018.
4. J. Becker, R. Knackstedt, and J. Pöppelbuß, "Developing Maturity Models for IT Management," Bus. Inf. Syst. Eng., vol. 1, no. 3, pp. 213–222, Jun. 2009.
5. D. Proença and J. Borbinha, "Maturity Models for Information Systems - A State of the Art," Procedia Comput. Sci., vol. 100, pp. 1042–1049, Jan. 2016.
6. P. Bahrs, "Adopting the IBM DevOps approach for continuous software delivery: Adoption paths and the DevOps maturity model," 2013.
7. I. Bucena and M. Kirikova, "Simplifying the devops adoption process," CEUR Workshop Proc., vol. 1898, 2017.
8. R. de Feijter, S. Overbeek, R. van Vliet, E. Jagroep, and S. Brinkkemper, "DevOps Competences and Maturity for Software Producing Organizations," in Enterprise, Business-Process and Information Systems Modeling, vol. 318, Springer, 2018, pp. 244–259.
9. Eficode Oy, "DevOps Quick Guides," Helsinki Finland, 2015.
10. S. Inbar, S., Sayers, Y., Pearl, G., Schitzer, E., Shufer, I., Kogan, O., & Ravi, "DevOps and OpsDev: How Maturity Model Works," Hewlett Packard Enterprise, 2013.
11. G. Menzel and A. Macaulay, "DevOps - The Future of Application Lifecycle Automation," Capgemini.Com, p. 24, 2015.
12. S. Mohamed, "DevOps shifting software engineering strategy-value based perspective," Int. J. Comput. Eng., vol. 17, no. 2, pp. 51–57, 2015.
13. S. I. Mohamed, "DevOps Shifting Software Engineering Strategy Value Based Perspective," IOSR J. Comput. Eng. Ver. IV, 2015.

AUTHORS PROFILE

Mohammad Zarour

Dr. Zarour holds a Ph.D. in Software Engineering (2009) from University of Quebec and master degree in Computer Science (1998) from University of Jordan. He is currently a faculty member in college of computer and information sciences (CCIS) at Prince Sultan University Riyadh, Saudi Arabia. He has more than 12 years of teaching experience in university and academic environment and also has several years of industry experience in information systems development and project management. His research interests include software process assessment and improvement, software quality, cost estimation, and web technologies. He has many peer-reviewed publications.

Norah Alhammad

Mrs. Norah has a master degree in software engineering from Prince Sultan University Norah has a Master degree in Software Engineering from Prince Sultan University. Her work focus in software development and DevOps. Currently, she is an Information Technology Project Manager in Semi Government Industry.

Mamdouh Alenezi

Dr. Alenezi is currently the Dean of Educational Services and the Chief Information & Technology Officer (CITO) at Prince Sultan University. Dr. Alenezi received his MS and Ph.D. degrees from DePaul University and North Dakota State University in 2011 and 2014, respectively. He has extensive experience in data mining and machine learning where he applied several data mining techniques to solve several Software Engineering problems. He conducted several research areas and development of predictive models using machine learning to predict fault-prone classes, comprehend source code, and predict the appropriate developer to be assigned to a new bug.

Khalid Al-Sarayreh

Dr. Al-Sarayreh is an associate professor of Software Engineering at the Prince Sultan University in Saudi Arabia. He has a Ph.D. degree in Software Engineering from the University of Québec in Canada. He also has a doctoral degree in Computer Information Systems, MSc in Computer Engineering (Embedded Systems) and BS degree in Computer Science from Jordanian Universities. He served during 25 years in both national and international institutions. With over 70 publications, his research interests include Software Quality Engineering, Software Quality Assurance using international standards, Software Requirements (Functional and Nonfunctional requirements), Software Measurement, Software Reuse, Software Engineering Standards (ECSS, IEEE, and ISO).