IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 August 2024, accepted 2 September 2024, date of publication 6 September 2024,
date of current version 19 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3455937

RESEARCH ARTICLE

SaaS Application Maturity Assessment Model

SAIQA ALEEM[1], (Senior Member, IEEE), RABIA BATOOL[2], SHAYMA ALKOBAISI[3], FAHEEM AHMED[4], AND ASAD MASOOD KHATTAK[2], (Senior Member, IEEE)

[1]Department of Physics and Computer Science, Faculty of Science, Wilfrid Laurier University, Milton, ON L9T 5E1, Canada
[2]College of Technological Innovation, Zayed University, Abu Dhabi, United Arab Emirates
[3]College of Information Technology, United Arab Emirates University, Al Ain, United Arab Emirates
[4]Department of Engineering, Thompson Rivers University, Kamloops, BC V2C 0C8, Canada

Corresponding author: Saiqa Aleem (saleem@wlu.ca)

This work was supported by the RIF Activity R17061 granted by the Zayed University, Abu Dhabi, United Arab Emirates.

ABSTRACT Software-as-a-service (SaaS), as a software delivery model, has received substantial attention from software providers and users alike. In recent years, it has become one of the most promising service delivery models in cloud computing. Many existing companies are transferring their business into the SaaS delivery model. Network vendors also migrate to a SaaS business model by offering on-demand remote IT support. This increasingly competitive landscape and the variety in markets have imposed many challenges for SaaS developers and vendors and made it difficult to find a consensus on the factors contributing to the positive performance of SaaS businesses. This paper thoroughly explains the critical success factors in the SaaS application development process. The proposed SaaS maturity model evaluates the organizations' current SaaS development methodology. The model's framework includes an assessment questionnaire, performance scale, and rating method adapted from the BOOTSTRAP algorithm. The assessment questionnaire collects information about the organization's current process, practices, and policies and calculates the organization's maturity level based on the responses. This study considers four dimensions to access the maturity level, i.e. design, architecture, business performance, and overall SaaS organization. Consequently, this work formulates a comprehensive and integrated strategy for SaaS application development maturity evaluation.

INDEX TERMS Software as a service, SaaS development performance, SaaS maturity assessment, SaaS design.

I. INTRODUCTION

The Software as a Service (SaaS) is a software delivery model in which third-party providers provide software as a service rather than as a product over the Internet for multiple users [1]. Services are installed, assembled, and maintained in the SaaS provider's systems. Users pay on the pay-per-use pricing model and get a flexible experience in terms of time and location of the access. It enables companies and organizations to use various IT services without the need to purchase, install, and maintain their IT infrastructure. They become free from most IT responsibilities of troubleshooting and maintaining the software. They can access services through the network from different vendors according to their business needs and pay the vendors per their usage. In the last few years, SaaS has been proven to be one of the most promising service delivery models in cloud computing [18]. According to Koraza [2], the market is expected to grow to over $230 billion by the end of 2024. The SaaS market will grow with an expected market value of over $900 billion by 2030 and grow at a CAGR (Compound Annual Growth Rate) of 18.7%. This is because SaaS providers are bringing nearly all functional extensions and add-ons to the application as a service, and SaaS solutions deliver better business outcomes than traditional software.

Organizations are adopting the SaaS model drastically [2]. There are many advantages associated with the SaaS delivery model for both consumers and providers. Providers require deploying only one application instance for multiple users. The improved hardware utilization rate reduces the overall application cost, attracting customers in small and medium enterprises [3]. It facilitates flexible subscriptions and access

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

to the latest innovations and functionality. It also supports upgrades without additional costs and provides reduced overheads, cost-effective infinite scalability, increased accessibility and productivity, higher-quality offerings at lower costs, and platform independence [4]. SaaS also integrates with other software used by businesses.

The SaaS architecture design, database partitioning, database architecture, scalability issues, user interface design, use of APIs and workflow differ from traditional applications. Its architecture should be able to fulfill functional as well as nonfunctional requirements. In the SaaS model, providers are also required to run and monitor the application on behalf of their customers. These unique features of SaaS require a higher level of quality than traditional software. However, the SaaS model has many advantages and associated challenges compared to on-premise software. For instance, multi-tenancy, which promises to provide a high degree of resource sharing among many tenants, also introduces many challenges during the design and operation of a SaaS system. Scalability, operational costs, flexibility, and security are critical for multi-tenant SaaS systems development, deployment, and management [5]. Unique requirements from the individual customer may increase the operating cost.

Moreover, the complexity of multi-tenant architecture can lead to poor performance and low resource utilization [6]. A different approach is required for the development of services on cloud computing than the conventional software development lifecycle. The unpredictable variation of individual tenants' performance can affect tenants' service level agreements. Serving tenants on a single instance and the impact of tenant customization on the whole system can lead to more complex processing [6]. Thus, SaaS vendors must consider an appropriate solution to abate the negative aspects of SaaS. Many researchers have explored SaaS key features [7], [8] and proposed frameworks for SaaS development [9], [10], [11].

The process Maturity model can be used to assess the maturity of different attributes. Software processes can be at any level of maturity and able to meet all necessary business requirements. Researchers have proposed a maturity model for software development [12] and maintenance [13]. Since SaaS is a different model than traditional software models, there is a need for a SaaS maturity model that should highlight all the features of SaaS design, architecture, business, and overall critical organizational factors. This paper proposes a five-level SaaS maturity model that will help assess the maturity of SaaS projects and provide improvement guidelines. Our software maturity model comprises assessment questionnaires about the critical factors identified from the literature and a rating methodology. These assessment questionnaires were used to collect the information and perform an empirical investigation on four perspectives of SaaS, such as design [15], architecture [14], business [16], and organization [17]. Our proposed SaaS maturity model will help SaaS vendors compete in the marketplace, improving customer satisfaction.

The SaaS maturity model has been applied to two SaaS development organizations, yielding results discussed in subsequent sections. The proposed maturity model will help organizations build the capacity to identify gaps and bottlenecks in their current processes. This proposed maturity model will also help SaaS development organizations improve their business model and determine their current and target positions, along with a roadmap to improve the current attitude towards the target position.

A. RESEARCH MOTIVATION

The growth of the SaaS market has many factors, such as the adoption of Artificial Intelligence, personalized experiences, focus on customer success, technological integration, and push for environmental, social and governance initiatives [18]. SaaS has become a desirable solution as compared to traditional on-premise software. It is an affordable solution for small and medium-sized companies. Being a SaaS company is no longer a competitive advantage, but it drags the vendor into a new competition. The rapid growth of the SaaS market does not seem to abate soon, which makes it very hard for the SaaS vendors to stand out and achieve substantial growth. This competition has increased customer expectations for quality and features and reduced switching costs. For customer retention and continued growth, there is a need for effective development and implementation strategies.

Fan et al. [19] highlighted many challenges SaaS vendors experience when providing software services. Software service provision can be costly, and as the customers are entirely dependent on the required service, they become more concerned about system reliability and availability. Multitenancy also introduced many complexities in SaaS design and development.

This highly competitive environment and complexities associated with SaaS development and maintenance motivate us to propose a maturity model that considers multiple dimensions of a SaaS product, such as architecture, design, business and organization, which will help to improve the SaaS quality and withstand the competitive environment for continued growth. The significant contribution of this work is to provide the first SaaS maturity model based on four dimensions that help streamline SaaS development activities such as concept planning, market research, feature planning technology stack, design architecture, backend and front end development; maintenance involves activities such as monitoring logging, bug fixing and updates, and scaling; growth processes such as user acquisition, customer retention, revenue optimization and expansion and may also facilitate stakeholders to make informed decisions.

II. LITERATURE REVIEW

A. SAAS DEVELOPMENT PROCESS

Most SaaS development on cloud computing requires a different approach than the conventional software development life cycle because SaaS development is a key feature of

the whole cloud project. In SaaS, each customer is unique, which requires unique software variation. Multi-tenancy and resource-sharing benefits in terms of higher scalability and resource utilization, but at the same time, they increase complexity in SaaS development [20]. To exploit the comparative advantages of the SaaS model efficiently and effectively, SaaS development companies must use a proper quality model to evaluate SaaS quality. Jagli et al. [21] presented the SaaS development life cycle in six stages, as shown in Figure 1. For high-quality SaaS development, La and Kim [22] proposed a systematic process with engineering instruction and highlighted the importance of commonality and variability modeling. Their developing process has eleven phases: requirement gathering, domain analysis, functional modeling, structure modeling, dynamic modeling, architectural design, UI design, Database design, implementation, testing, and deployment.

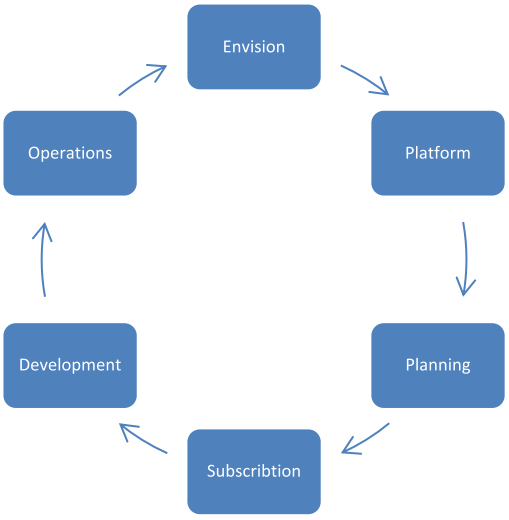

FIGURE 1. SaaS development Lifecycle.

Similarly, Espadas et al. [23] analyzed the effects of SaaS on software development phases. They also redefined the five stages of the SaaS development process. These five stages include the requirement, analysis, design, implementation, and reusability. They also provided guidelines for application development in the SaaS environment. Moreover, Mandal et al. [24] proposed SaaS architecture for data-centric cloud applications, which eases application maintenance and enhances overall application flexibility. Their proposed architecture consists of multiple layers which can interact with each other. They also performed a comparative analysis of the existing SaaS architectural framework based on its key characteristics. Espadas et al. [25] also proposed an architecture for developing, deploying, and managing SaaS applications. Many researchers proposed a framework to design and implement SaaS applications effectively. Guo et al. [9] provided an SOA-based framework comprising multi-tenancy standard services and principles for effective isolation among tenants. This framework would allow developers to focus on business logic instead of multitenancy requirements. EasySaaS is another SaaS development framework proposed by Tsai et al. [10]. EasySaaS facilitates building SaaS in two ways. The tenant can publish application requirements and test scripts, and SaaS developers can customize them according to the tenant's requirements, or tenants can compose an application using the templates provided. This also supports testing in the development process. Mietzner et al. [26] proposed a package format for composite configurable SaaS application packages. They focused on the reusability of SaaS using service component architecture. Moreover, many researchers also focused on the individual phase of the SaaS development life cycle, such as platform [27], [28], [29], [30] subscription [31], [32], [33] planning [34], [35] and architecture [36], [37].

B. SOFTWARE DEVELOPMENT MATURITY MODEL

The software development process contains software product development, deployment, maintenance, organizational policies, organizational structures, human activities, and the functionalities and technologies used in the process [38]. Software development requires methodology, tools, and a group of people dedicated for some time. It is the process which matures with the maturity of the organization. The maturity level of any organization can be measured by its capability to define, manage, measure, and control the software development process.

Researchers have been working on an assessment of the organization's software process maturity. Assessment methodologies should provide complete qualitative and quantitative information to diagnose and improve processes. Some well-known international organizations, such as the International Standards Organization (ISO), the Software Engineering Institute (SEI), the International Electro-Technical Commission (IEC), and the European Software Institute (ESI) have defined standards for software process assessment. The level maturity model was proposed by the Software Engineering Institute (SEI) [39], which defines key performance areas from the initial state to a higher level of optimization. This model has been adopted by most organizations in the software industry to develop and define their software development process and prioritize their improvement efforts. Initially, the process is ad hoc, and success depends on quality people. The second level is repeatable, in which project management processes are established to track cost, schedule, and functionality. In the third named defined level of CMM, the software process activities are documented, standardized, and integrated into all processes for the organization. The organization's approved standard software process is used to develop and maintain all projects. At the fourth level of maturity, detailed measurements of the software process and product quality are collected and controlled. At the fifth and optimizing level, continuous process improvement is enabled by quantitative feedback from the process and piloting innovative ideas and technologies. CMMI models have evolved the Capability Maturity

Model (CMM) concept, which integrates systems engineering, software engineering, product and service development, supplier sourcing, and traditional CMM concepts such as process management and project management. Its architecture included a set of 22 process areas arranged in two representations: one per stage. The areas are grouped into five levels of maturity [40]. Bootstrap [41] is another software process assessment and improvement methodology that includes a guided assessment process, maturity and capability determination instruments, guidelines for process improvement, an assessor training program and licensing policy, and computer-based tools and a European database that support the consultancy activities. It comprises five levels but divides the process area into technology, organization, and methodology.

SPICE (Software Process Improvement and Capability dEtermination) [42] is an international framework for the assessment of software processes which can be used in process improvement and process capability determination. It has six maturity levels and a set of nine documents: the first six mainly focus on addressing various aspects related to process assessment, 7 and 8 address process assessment for process improvement and capability determination, and 9 acts as a vocabulary [43]. The family of ISO-9000 standards supports setting up a quality management system within an organization to develop and maintain the software and many other purposes. ISO-9000-3 can apply ISO-9001 to software supply, development, maintenance, and installation. It also presents guidelines for documentation, responsibility, corrective actions, and software development audits [44]. ISO-12207 [45] provides a framework for improving software engineering and management by categorizing the broader classes, such as primary life cycle, support lifecycle, and organizational life cycle processes. ISO/IEC 12207:2008 provides a well-defined terminology that the software industry can reference.

The abovementioned methods focus mainly on engineering process assessment, specifically for software development activity. As discussed earlier, the SaaS development and maintenance process differs from traditional software development and maintenance. So, these methods cannot be directly applied to assess SaaS design, architecture, business, and organization.

C. SAAS ASSESSMENT AND MATURITY MODELS

SaaS is a complex business model with many roles, such as customer, developer, vendor, etc. It is very essential to assess and manage its quality based on requirements. Current application service providers also require a maturity model to migrate their systems to the SaaS platform. Researchers agreed that the SaaS maturity model can be presented incrementally. The vendor may not be required to reach the highest level of maturity to fulfill their business requirements. They can decide their level based on business needs, the targeted customers, architectural characteristics, and financial and operational considerations [37]. SaaS maturity can be considered a continuum between isolated data and code on one end and shared data and code on the other. The position of this continuum can be decided based on the business architectural and operational model [46].

Many researchers have proposed a maturity model for SaaS applications. Microsoft introduced the first widely published incremental SaaS maturity model for SaaS application architecture using three key attributes: configurability, multi-tenancy, and scalability. Their model has four levels. At Level 1, SaaS does not have scalability, multi-tenancy, or configuration. Level 2 is configurable, and level 3 is configurable and multi-tenant. The fourth level adds scalability to the 3rd level. Despite the well-defined incremental structure, measuring Application service provider (ASP) vendors' availability is still ambiguous due to the lack of detailed concepts. Forrester's research [47] also proposed a SaaS application evaluation model. It is similar to Microsoft's maturity model but is divided into six levels, from traditional ASP to the SaaS service model. They did not discuss the incremental process of evaluation. Kang et al. [48] defined a general practical SaaS maturity model with two important axes: service component and maturity level. Four levels in their maturity model are Ad Hoc Level, Standardization Level, Integration Level and Virtualization Level. Each level acts as a foundation to evolve to the next level. Their maturity model can be adopted incrementally to migrate the current ASP service to SaaS. Chen [49] proposed a SaaS evaluation model based on five quality matrices and four roles: service customer, service provider, service broker and the service evaluation tool. Lee et al. [8] also presented a SaaS application evaluation model in which they added new features such as reliability, efficiency, scalability, reusability, and availability. They also assess their models regarding correlation, consistency, and discriminative power, as defined in IEEE 1061.

Based on the importance of configurability of the user interface and its modification effects on the logic layer, Wang et al. [50] added another dimension named configuration level to the SaaS maturity model. Hence, their proposed model is a two-dimensional maturity model. One dimension is the original Levels 2, 3, and 4; the new dimension is the "Configuration" level.

Some researchers presented their applications at different SaaS maturity levels. Hudli et al. [51] presented level-4 SaaS applications for the healthcare industry and incorporated the application's security. Similarly, Li et al. [52] designed a credit bank information system (CBIS) based on the SaaS Level 3 maturity model.

In the SaaS development area, no studies have been published that directly address the issue of process improvement and assessment. Most of the related work proposed a maturity model for SaaS architecture and did not consider key features of SaaS, such as security, availability, quality of service, etc [53]. They also do not focus on other dimensions, e.g. business, organization, etc [54]. The research gap in this area motivated the authors to propose the SaaS Maturity

Model, which considers all the key features of multiple dimensions.

SaaS provides an integrated solution over the Internet, which may require weekly innovation and changes. SaaS vendors do not only deal with developing but also running and monitoring the software on behalf of their customers. SaaS development requires a different approach than the traditional software development lifecycle. It should be able to fulfill the varying requirements of multiple users while maintaining its availability and security. The SaaS development process requires a solid set of best development practices based on these requirements. This SaaS maturity model aims to create a comprehensive set of key practices to evaluate SaaS development and maintenance processes. It describes the assessment methodology for SaaS requirements analysis, architecture, design, business and organization and measures the current level of maturity for any SaaS application. To our knowledge, it is the first study of its kind.

III. SAAS MATURITY MODEL

SaaS provides an integrated solution over the Internet, which may require weekly innovation and changes. SaaS vendors do not only deal with developing but also running and monitoring the software on behalf of their customers. SaaS development requires a different approach than the traditional software development lifecycle. It should be able to fulfill the varying requirements of multiple users while maintaining its availability and security. The SaaS development process requires a solid set of best development practices based on these requirements. This SaaS maturity model aims to create a comprehensive set of key practices to evaluate SaaS development and maintenance processes. It describes the assessment methodology for SaaS requirements analysis, architecture, design, business and organization and measures the current level of maturity for any SaaS application. To our knowledge, it is the first study of its kind.

A. THE GENERAL SCOPE OF THE SAAS MATURITY MODEL

The SaaS assessment is essential to improve an organization's development and maintenance practices. A maturity model had two main objectives. First, it designs a method to perform the assessment, and second, it provides guidelines to improve the whole process. For a SaaS system, it seems impossible to make an efficient and effective development plan without detailed assessment results. Figure 2 shows a detailed framework for SaaS organizations to practice SaaS development and maintain the process. Our SaaS maturity model considers all critical development and maintaining activities to develop the framework, which consists of maturity levels and surveys for practice assessment. The maturity model assesses the current level of key activities in an organization.

B. CONFIGURATION OF SAAS MATURITY MODEL

The configuration of our SaaS maturity model has twenty-three key activities. The hierarchy of maturity model is described in Table 1. These twenty-three key activities are

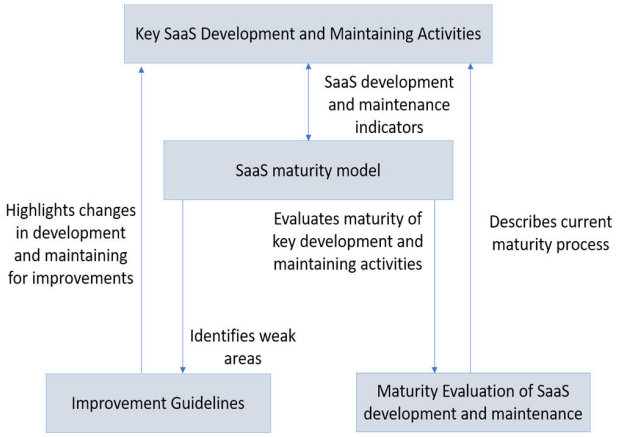

FIGURE 2. Scope of SaaS maturity model.

divided into four dimensions, i.e. SaaS application design, improved SaaS development process, SaaS application business performance, and SaaS organizational performance. SaaS application design includes domain analysis, design pattern, design for failure, design for parallelism, and QoS Differentiation. The SaaS development process dimension covers architectural concerns: customization management, scalability management, MTA (Multi-Tenancy Architecture), Security, Integration Management, and Fault tolerance and recovery management. The authors conducted a literature review and empirical investigation from a developer perspective and selected these six key factors essential for the SaaS development process. The SaaS application business performance dimension includes six important process activities: Monitoring management, Marketing strategies, Innovation, Risk mitigation, Customer satisfaction, Business collaboration, and Competitors. The selected key features in the SaaS application business Performance dimension are significant for the business performance of any SaaS application. The SaaS organizational performance dimension includes important factors for any SaaS organization. These factors are Organizational learning, change management, conflict management, organizational structure and organizational culture. The author performed an empirical investigation to measure the impact of identified organizational factors on SaaS organizations. The empirical studies carried out to identify key factors in the SaaS development process and their presence in the literature motivated the inclusion of twenty-three key process activities under the four SaaS maturity model dimensions.

A description of these key process activities is given below.

1) DOMAIN ANALYSIS

Domain analysis involves identifying standard and variable features in scoping and modeling SaaS. A good SaaS design must have a domain analysis process to define a set of common definitions, domain classification, domain boundaries, domain models, design artifacts, and design guidelines.

TABLE 1. Configuration of the software maturity model.

Dimensions	Activity ID (AID)	SaaS development and maintaining Activities
SaaS application design	1	Domain analysis
	2	Design pattern
	3	Design for failure
	4	Design for parallelism
	5	QoS Differentiation
SaaS development process	6	Customization management
	7	Scalability management
	8	MTA
	9	Security management
	10	Integration management
	11	Fault tolerance and recovery management
Business performance	12	Monitoring management
	13	Marketing strategies
	14	Innovation
	15	Risk mitigation
	16	Customer satisfaction
	17	Business collaboration
	18	Competitors
Organizational performance	19	Organizational learning
	20	Change management
	21	Conflict management
	22	Organizational structure
	23	Organizational culture

2) DESIGN PATTERNS

Design patterns provide an abstract solution to recurrent problems in a domain. They can help improve the scalability, interoperability, multi-tenancy, flexibility, manageability, and portability of the SaaS system.

3) DESIGN FOR FAILURE

Deciding on application behavior in case of failure is called failure analysis. Advanced features of SaaS increase the chances of failure, so SaaS applications must be designed and developed, assuming they tend to fail often. Design for failure is a key feature of SaaS software design and plays a significant role in the success of SaaS vendors.

4) DESIGN FOR PARALLELISM

The main objective of parallel computing is to increase the available computation power for faster application processing. In SaaS, where heavy applications serve thousands of users simultaneously, ''Design for parallelism'' is considered an essential factor.

5) QOS DIFFERENTIATION

The difference in tenant service quality is called QoS differentiation [55]. Any SaaS system should be designed to provide QoS differentiation and to satisfy the provider's need for the least resource cost and best system performance.

6) CUSTOMIZATION MANAGEMENT

For a multi-tenant system, each tenant has different requirements. To meet the requirements of different tenants, a successful SaaS application must support a customization feature which allows each user to customize it based on his requirements.

7) SCALABILITY MANAGEMENT

Any system which can handle an increasing quantity of work by maintaining stable performance by adding new resources is called a scalable system. In a multi-tenant system, each tenant can have hundreds of thousands of users, resulting in many simultaneous accesses from users. Thus, a SaaS system should be able to provide scalable performance to millions of users.

8) MTA

The ability of a SaaS application to serve multiple tenants using a single server and service instance in such a way that every tenant feels like a dedicated server is called multitenancy. Multi-tenancy is a crucial feature of SaaS software and is one of the measures for the success of SaaS vendors.

9) SECURITY MANAGEMENT

Resource sharing in multi-tenant SaaS applications increases the security and privacy concerns of outsourced cloud-hosted assets; hence, security is one of the key features of SaaS applications.

10) INTEGRATION MANAGEMENT

SaaS applications must integrate with other services and applications to get the desired business functionality. So, SaaS systems must support integration with other SaaS and on-premises systems.

11) FAULT TOLERANCE AND RECOVERY MANAGEMENT

Fault tolerance is the feature that helps the system operate continuously in case of any failure. To achieve robustness, failure should be assessed and handled effectively in SaaS.

12) MONITORING

Monitoring is one of the significant challenges in SaaS design and development. To handle the diversity of SaaS systems, the demand for monitoring grows and brings new targets, methodologies, and techniques [56].

13) MARKETING STRATEGY

SaaS marketing is critically different from every other type of marketing. SaaS provisioning is a cost-efficient business, but marketing and sales costs have been identified as key factors in the profitability of SaaS providers. To enhance the SaaS market growth, SaaS providers need to apply effective marketing strategies.

14) INNOVATION

Innovation at both technical and non-technical levels is important to enhance the quality and performance of the business. SaaS service can be considered an innovation for both providers and customers. In the SaaS business model,

the innovation style has modified from local innovation of a software user to collective innovation of an entire system of users and software.

15) RISK MITIGATION

Risk mitigation is a systematic procedure for reducing risk. SaaS is the most used cloud service model and requires security practices because customers lose control and SLAs do not provide details on security implementation.

16) CUSTOMER SATISFACTION

Customer satisfaction is a marketing term that measures how products or services from providers meet customers' expectations. Customer satisfaction is also a key metric for SaaS companies because acquiring new business is more expensive and time-consuming than building upon an existing relationship.

17) BUSINESS COLLABORATION

Collaboration allows companies to aggregate competencies with other companies by focusing on their skills to increase the value of their business. Collaborative SaaS is a new area.

18) COMPETITORS

Competitor analysis helps assess the strengths and weaknesses of current and potential competitors and implement effective strategies to improve competitive advantage. SaaS vendor must analyze their market competition and plan accordingly.

19) ORGANIZATIONAL LEARNING

Organizational learning is the process of creating, retaining, and transferring knowledge within an organization to enhance its collective ability to accept and respond to internal and external changes. For SaaS applications, it is crucial to achieve business performance and customer satisfaction.

20) CHANGE MANAGEMENT

Change management is a structured approach to preparing, equipping, and supporting individuals to adopt and implement change successfully. It allows them to capture the adoption contribution that drives organizational success and outcomes. Change influences the whole organization, and resistance to it is natural. Therefore, change management is an essential activity in SaaS organizations.

21) CONFLICT MANAGEMENT

Conflict within groups refers to disagreements and friction among team members. Conflict can be highly constructive as in the absence of conflicts, teams may not realize existing inefficiencies; therefore, for any SaaS organization, conflicts should be managed appropriately to achieve their benefits.

22) ORGANIZATIONAL STRUCTURE

Organizational structure can be considered the organization's anatomy, directly affecting the behavior of organization members. SaaS organizations should adopt their structure based on their requirements, skills, and originality.

23) ORGANIZATIONAL CULTURE

The culture of any organization directly affects its employee satisfaction, organizational success, productivity, and performance. SaaS managers should play a role in the development and maintenance of organizational culture as it directly influences the behavior of its employees, and it is tough to change once developed.

These twenty-three important key practices are the foundation of the assessment questionnaires, which consist of ''statements''. These statements describe the effectiveness of the particular activities as they contribute to SaaS development and management.

C. THE FRAMEWORK OF THE SAAS MATURITY MODEL

Ranking is an essential assessment methodology for defining maturity level. An organization's ranking can be measured by calculating its progress against the scale of defined levels based on specific Key Process Areas (KPAs). The ranking has been used in many well-known software process assessment models, such as CMMI [57], SPICE [42], and BOOTSTRAP [41]. We also defined the level of maturity of our proposed maturity model using ranking. These levels are ''Ad-Hoc'', ''Opportunistic'', ''Consistent'', ''Organized'', and ''Optimized''. An assessment questionnaire and a ranking method are developed for each maturity level. The questionnaire is composed of multiple statements for each level of maturity. The extent to which respondents agree to these statements is used to calculate the maturity level of the organization's SaaS development process. Assessment questionnaires for each maturity level in this study are designed and written specifically for the SaaS maturity model.

This study is designed to find how SaaS development process activities/areas are performed during the SaaS project development life cycle. It is a comprehensive strategy to evaluate the maturity level of current SaaS Development Process Areas (SDPAs) in the SaaS project.

Table 2 shows the SaaS maturity model. The maturity of each level is defined by the set of statements for each activity used in this study. Each maturity level and process activity has a different number of statements. The following subsection describes the features of SaaS development organizations. Each SaaS organization will be described in terms of the SaaS development process activities maturity scale and the measuring instrument designed for the SaaS maturity model.

Level 1: Ad-Hoc

The initial level of the SaaS maturity model is ''Ad-Hoc''. This level shows that SaaS organizations are unstable and organized enough to develop SaaS projects. At the ad-hoc level, there is a deficiency in understanding the best practices for SaaS development, and the organization does not use specified software engineering practices. The organization

TABLE 2. SaaS maturity model framework.

	Number of Maturity Levels statements					
SaaS development process activities and number of statements in questionnaire	**Software Process development Activities**	**Ad-hoc**	**Opportunistic**	**Consistent**	**Organized**	**Optimized**
	Domain analysis	3	2	3	3	3
	Design pattern	3	2	2	2	1
	Design for failure	3	2	2	3	2
	Design for parallelism	2	2	1	1	1
	QoS differentiation	1	2	1	1	2
	Customization management	1	1	3	3	1
	Scalability management	1	1	1	1	1
	MTA	1	1	2	2	1
	Security management	1	1	2	3	1
	Integration management	1	2	1	2	3
	Fault tolerance and recovery management	1	2	2	2	2
	Monitoring management	2	1	1	5	3
	Marketing strategies	2	2	2	3	2
	Innovation	1	4	2	4	1
	Risk mitigation	1	2	2	3	3
	Customer satisfaction	1	1	3	2	4
	Business collaboration	1	2	1	2	2
	Competitors	1	1	1	4	1
	Organizational learning	2	2	1	1	2
	Change management	1	1	1	3	1
	Conflict management	2	1	1	2	3
	Organizational structure	2	2	2	2	2
	Organizational culture	1	2	1	1	2
	Total	**35**	**39**	**38**	**55**	**44**

develops various SaaS projects independently and does not follow any defined procedure for asset reusability for different projects. At this level, the organization does not have enough technical resources and skills to complete the project properly, and there is no evidence of following any requirements or management and development methodology. The organization neither performs domain, market and competitor analysis nor manages change and conflicts within the team. The following assessment questionnaire shows the maturity of SaaS development process activities in a SaaS development organization at Level 1.

SPDA 1.1. Domain analysis

1.1.1 There are no domain experts to perform domain analysis.
1.1.2 There is no evidence of any Domain scoping and domain modeling for the SaaS project.
1.1.3 There is no clear definition of SaaS variability to make SaaS processes and instructions more feasible and effective in designing reusable services.

SDPA 1.2. Design pattern

1.2.1 There is no evidence of design pattern in SaaS design.
1.2.2 No database schema pattern exists.
1.2.3 No customization pattern exists.

SDPA 1.3 Design for failure

1.3.1 There is no consideration of design for failure.
1.3.2 SaaS applications are not able to adopt changes in infrastructure without downtime.
1.3.3 Threads are not able to resume reboot in case of any failure.

SDPA 1.4 Design for Parallelism

1.4.1 There is no design for parallelism in SaaS project design.
1.4.2 The design team does not consider leveraging multi-core processors to speed up computationally intensive applications.

SDPA 1.5 QoS differentiation

1.5.1 The design team does not consider QoS differentiation.

SDPA 1.6 Customization management

1.6.1 No process exists for the customization to meet multiple tenants' needs.

SDPA 1.7 Scalability management

1.7.1 There is no support for scalability in SaaS architecture.

SDPA 1.8 MTA

1.8.1 Each customer runs a different instance of the software.

SDPA 1.9 Security management

1.9.1 There is no evidence of security provision in SaaS architecture.

SDPA 1.10 Integration management

1.10.1 There is no support for integration in SaaS architecture.

SDPA 1.11 Fault tolerance and recovery management

1.11.1 There have been no efforts in SaaS architecture to address fault tolerance and recovery management.

SDPA 1.12 Monitoring management

1.12.1 There is no evidence of monitoring of SLA for SaaS applications.

1.12.2 SLA does not specify the guidelines monitors use to conduct any monitoring session.

SDPA 1.13 Marketing Strategies

1.13.1 The project team has not developed any marketing strategy for the SaaS project.

1.13.2 There is no evidence that the team performs market analysis of the SaaS type and the target Audience.

SDPA 1.14 Innovation

1.14.1 There is no evidence of any research and development component working on innovation.

SDPA 1.15 Risk mitigation

1.15.1 Management does not practice risk mitigation at any stage of the SaaS development process.

SDPA 1.16 Customer satisfaction

1.16.1 Customer satisfaction is not considered a key metric of SaaS project success.

SDPA 1.17 Business Collaboration

1.17.1 The management team does not understand the importance of business collaboration for the SaaS project.

SDPA 1.18 Competitors

1.18.1 Management and development teams do not consider competitors when deciding about a SaaS project.

SDPA 1.19 Organizational learning

1.19.1 Organizational learning is not considered important.

1.19.2 The organization does not learn from its experience.

SDPA 1.20 Change management

1.20.1 The management team does not consider organizational change management complementary to project management.

SDPA 1.21 Conflict management

1.21.1 Teams are engaged in relationship conflict, are overloaded cognitively and are unable to complete team tasks.

1.21.2 Conflict is dysfunctional and personally oriented.

SDPA 1.22 Organizational structure

1.22.1 The roles and responsibilities of individuals and groups are not well defined.

1.22.2 The roles and responsibilities of individuals and groups are not adequately documented.

SDPA 1.23 Organizational culture

1.23.1 The management team does not focus on improving organizational culture.

Level 2: Opportunistic

"Opportunistic" is the second maturity level of our SaaS maturity model. At this level, the management and development team understand and realize the importance of best SaaS development and specified software engineering practices and want to adopt them in the SaaS project. The development team shows interest in acquiring skills and knowledge for the required development methodologies. The management team and development team support the idea of design patterns, parallel processing, and integration in SaaS design and architecture. They also understand the impact of change and conflict management on the organization's culture and the performance of the SaaS application project. However, there is a lack of resources, policies and systematic planning to practice the best software engineering techniques for SaaS development. There is also the absence of guidelines for conflict and change management, marketing and competitor analysis, and business collaboration. At this level, SaaS architecture allows minimal customization through configuration. This level is opportunistic because the organization focuses on understanding the best practices to develop high-quality, successful SaaS projects. At this level, teams understand the importance of adopting best practices for SaaS development, and organizations seek the opportunity to build their understanding of best practices and acquire enough resources and skills to move to the next level. The following assessment questionnaire shows the maturity of SaaS development process activities in a SaaS development organization at Level 2.

SDPA 2.1 Domain analysis

2.1.1 Informal attempts are made to perform commonalities and variability of components and services.

2.1.2 There is little evidence of domain scoping and modeling.

SDPA 2.2 Design pattern

2.2.1 Project managers and team leaders understand the importance of design patterns in SaaS design and development.

2.2.2 Project managers and team leaders believe design patterns would help capture the best application design practices and build reliable, compliant, reusable SaaS products.

SDPA 2.3 Design for failure

2.3.1 Designers recognize the need for design for failure during the design phase of SaaS development.

2.3.2 Designers are working to make the system fault-tolerant.

SDPA 2.4 Design for Parallelism

2.4.1 The SaaS management team believes that GPU can make a significant difference in the quality of services.

2.4.2 The design team believes that leveraging multi-core processors can be used to speed up computationally intensive applications.

SDPA 2.5 QoS differentiation

2.5.1 SaaS designers think they need to compose services with different QoS values to meet end-users' different multidimensional QoS constraints for the SaaS.

2.5.2 The management team agrees that Different clients may have different needs and willingness to pay for system performance.

SDPA 2.6 Customization management

2.6.1 The architecture allows minimal customization through configuration.

SDPA 2.7 Scalability management

2.7.1 Project managers and team leaders understand the importance of scalability in SaaS architecture.

SDPA 2.8 MTA

2.8.1 The development team members are acquiring knowledge and skills to support multi-tenancy in SaaS architecture.

SDPA 2.9 Security management

2.9.1 The development team understands the importance of security but has no proper plan to add security to the architecture.

SDPA 2.10 Integration management

2.10.1 The development team recognizes the importance of integrating SaaS with on-premise and other SaaS applications.

2.10.2 The development team recognizes the importance of integration support in the user interface and business logic layers.

SDPA 2.11 Fault tolerance and recovery management

2.11.1 The developer introduced fault tolerance in SaaS architecture.

2.11.2 The development team is acquiring resources and skills to develop a proper fault tolerance and recovery management strategy.

SDPA 2.12 Monitoring management

2.12.1 The management and development team recognizes the importance of monitoring SaaS applications.

SDPA 2.13 Marketing Strategies

2.13.1 The project team agrees that a marketing strategy is essential for a SaaS project.

2.13.2 There is a lack of resources to make a marketing strategy.

SDPA 2.14 Innovation

2.14.1 No well-defined policy for research and development (R&D) has been established.

2.14.2 Innovative ideas are considered necessary for new SaaS projects.

2.14.3 The development team occasionally studies and reviews development updates and searches for innovative ideas.

2.14.4 There is no well-developed SaaS business model to capture value from innovation.

SDPA 2.15 Risk mitigation

2.15.1 Management understands its responsibility to protect valuable business data and systems.

2.15.2 The management team is committed to acquiring knowledge and resources related to SaaS development risk.

SDPA 2.16 Customer satisfaction

2.16.1 Management understands the importance of customer satisfaction.

SDPA 2.17 Business Collaboration

2.17.1 Management believes that business collaboration is vital for SaaS projects.

2.17.2 The management team does not make efforts to collaborate with other businesses.

SDPA 2.18 Competitors

2.18.1 The management and development team is committed to defining a proper strategy to perform competitor analysis.

SDPA 2.19 Organizational learning

2.19.1 Project managers and development team agrees that organizational learning is fundamental to stand out in the competitive SaaS market.

2.19.2 learning is considered important for the resource management of new ventures in the SaaS industry.

SDPA 2.20 Change management

2.20.1 The organization understands the impact of change management on the performance of the organization in managing SaaS application projects.

SDPA 2.21 Conflict management

2.21.1 The development and management team understands that conflicts are affecting team performance.

SDPA 2.22 Organizational structure

2.22.1 The management team believes well-defined roles and responsibilities are important for any SaaS project.

2.22.2 The project manager and the team lead agree that an organized team can perform its assigned tasks more effectively.

SDPA 2.23 Organizational culture

2.23.1 The management team agrees that organizational culture indirectly influences inter-organizational relationships.

2.23.2 Team members are not satisfied with the current culture of the organization.

Level 3: Consistent

Organizations at this level consistently try to define policies and strategies for SaaS development projects. This

interest in defining a strategic plan shows that the organization aims to develop good-quality SaaS products. At this level, the organization is devoted to developing a SaaS architecture with all quality attributes and considers domain, market and competitor analysis mandatory activities. The management team organizes learning workshops related to required development methodologies. The organization is trying to manage associated risk, continual monitoring, and organizational conflicts in an effective way. The management and development teams have acquired enough training in development methodologies, and the development team is trying to define design patterns and strategies for fault tolerance and security. They are also involved in defining R & D policies. The organization occasionally performs market and competitor analysis to make required changes and is making strategies for managing these changes. The organization is trying to be consistent in its SDPAs for the development process. The following set of statements must be satisfied by an organization at Level 3.

SDPA 3. 1 Domain analysis

3.1.1 Designers are working on the proper understanding of domain analysis.
3.1.2 Requirement Normalization is performed to perform commonality and variability analysis.
3.1.3 There are clear guidelines for identifying SaaS variability.

SDPA 3.2 Design pattern

3.2.1 The development team is making efforts to define design patterns for recurrent problems.
3.2.2 Design patterns are used to speed up the development process of SaaS applications.

SDPA 3.3 Design for Failure

3.3.1 A resource requirement calculator is used to identify the requirements for each additional new tenant.
3.3.2 A well-defined procedure is used to identify and handle complete multi-tenancy requirements.

SDPA 3.4 Design for Parallelism

3.4.1 The design team has the resources and technical knowledge to introduce parallelism in a SaaS project.

SDPA 3.5 QoS differentiation

3.5.1 The management and development team are working to offer differentiated services rather than having a one-size-fits-all approach.

SDPA 3.6 Customization management

3.6.1 Customization points were modeled, and dependencies between different variation points were explicit before implementation.
3.6.2 There is a method to validate each tenant's choice for each possible error.
3.6.3 Standard specification techniques are used to specify components in the database.

SDPA 3.7 Scalability management

3.7.1 The design team is committed to developing scalability guidelines.

SDPA 3.8 MTA

3.8.1 The application architecture includes multi-tenancy support.
3.8.2 Multi-tenant support can be applied to a multi-tenant SaaS model's application, middleware, virtual machine, and operating system layers.

SDPA 3.9 Security management

3.9.1 The development team is acquiring resources and skills to develop a proper strategy for SaaS Security.
3.9.2 The development team is considering security concerns such as data security, application security, and deployment security in SaaS architecture.

SDPA 3.10 Integration management

3.10.1 The development team is trying to add integration support and a single sign-on feature in the user interface.

SDPA 3.11 Fault tolerance and recovery management

3.11.1 Fault detection algorithms detect faults in the host, object, and process.
3.11.2 There is an explicit code in the application that retries failed transactions.

SDPA 3.12 Monitoring management

3.12.1 The management and development team is working on defining policies for monitoring SaaS applications.

SDPA 3.13 Marketing Strategies

3.13.1 A marketing team is needed to create a marketing strategy.
3.13.2 The marketing team occasionally performs market reviews and development updates.

SDPA 3.14 Innovation

3.14.1 The management team believes that R&D investment will yield positive results soon.
3.14.2 The management team is in the process of defining an R&D policy.

SDPA 3.15 Risk mitigation

3.15.1 Strategies are developed to manage risks related to development strategy, staffing, budgeting and scheduling, and security.
3.15.2 The management team can perform reactive risk management.

SDPA 3.16 Customer satisfaction

3.16.1 Management is trying to improve customer satisfaction levels by improving the quality of services.
3.16.2 Management is trying to gain customer loyalty.
3.16.3 Feedback from the customer is helping to find the customer's problem.

SDPA 3.17 Business Collaboration

3.17.1 The management team occasionally collaborates with other businesses.

SDPA 3.18 Competitors

3.18.1 A competitor analysis strategy is developed during the pre-production phase.

SDPA 3.19 Organizational learning

3.19.1 1 Management wants to organize training programs for their employees.

SDPA 3.20 Change management

3.20.1 The project team is committed to improving change management for SaaS project performance.

SDPA 3.21 Conflict management

3.21.1 The management team is trying to design conflict management policy to handle conflicts in project teams.

SDPA 3.22 Organizational structure

3.22.1 The management team defines roles based on the current projects in the organization.

3.22.2 Organization structure has optimum ratios of supervisors/managers to subordinates.

SDPA 3.23 Organizational culture

3.23.1 A management team is essential to establishing and maintaining an organization's culture.

Level 4: ***Organized/Predictable***

The fourth maturity level of the SaaS maturity model has been defined as ''organized or predictable''. The organization has successfully developed policies and guidelines for SaaS development activities at this level. The development team has gained all the required resources and skills. The development team uses design patterns for recurrent problems in SaaS development and provides integration support for on-premise and other SaaS applications. The team performs market and competitor analysis regularly and uses innovative ideas for development. Domain analysis is considered mandatory during the pre-production phase of the project. The architecture supports all required customization, scaling, and security requirements.

Moreover, the SaaS project can achieve service differentiation without losing the critical benefits of multi-tenancy. At this level, management supports positive and constructive conflicts in the organization. Once developed, SaaS projects can satisfy customers, and their revenue model fits into the organization's financial model. The developed project is mature enough to handle any runtime fault without affecting the service continuity. Overall, all the SDPAs at this level are streamlined, quantifiable, and well-documented for any SaaS project. In the following measuring instrument, the set of statements describes an organization's maturity level at level 4.

SDPA 4.1 Domain analysis

4.1.1 Domain analysis is considered mandatory during the pre-production face of the SaaS project.

4.1.2 The degree of commonality determines the applicability of components or services.

4.1.3 Developed components are used widely due to domain analysis.

SDPA 4.2 Design pattern

4.2.1 Multiple design patterns have been developed to address recurrent problems in SaaS.

4.2.2 Different strategies are used for scaling a shared database versus scaling dedicated databases.

SDPA 4.3 Design for Failure

4.3.1 Fault modeling and fault injection are part of a continual release process.

4.3.2 SaaS providers are running the same service on multiple instances to avoid failure.

4.3.3 Application components are designed to be deployed across redundant cloud components with no common point of failure.

SDPA 4.4 Design for Parallelism

4.4.1 Designers have efficient design systems for task and data parallelism.

SDPA 4.5 QoS differentiation

4.5.1 SaaS can achieve service differentiation without losing the key benefits of multi-tenancy.

SDPA 4.6 Customization management

4.6.1 The architecture supports all required customization.

4.6.2 Data field, process, service, and interface customization are supported to meet the requirements of individual tenants.

4.6.3 Component information is stored in an ontology database, which is used for searching, discovery, and reasoning.

SDPA 4.7 Scalability management

4.7.1 The architectures allow applications to scale out quickly.

SDPA 4.8 MTA

4.8.1 Multi-tenancy improves the utilization rate of hardware resources, and software deployment and maintenance become easy.

4.8.2 There are defined guidelines that exist to handle multi-tenancy requirements.

SDPA 4.9 Security management

4.9.1 The security feature is successfully added to SaaS architecture.

4.9.2 SaaS application fulfilled run time emerging security requirements of old and new tenants.

4.9.3 A defined set of guidelines for the security approaches incorporated in SaaS applications exists.

SDPA 4.10 Integration management

4.10.1 The development team provides user interface and business logic layers integration support.

4.10.2 There is a single sign-on feature in the user interface.

SDPA 4.11 Fault tolerance and recovery management

4.11.1 One or two replicas are assigned to mask the failure of any server.

4.11.2 Reactive Fault tolerance techniques are used to address the failure.

SDPA 4.12 Monitoring management

4.12.1 Metering and monitoring are done from both data and execution standpoints.

4.12.2 SaaS application has appropriate monitoring metrics for different customers.

4.12.3 SLA specifies the guidelines used by monitors to conduct any monitoring session.

4.12.4 Monitoring metrics provide information in the same terms as the SLA is specified.

4.12.5 Monitoring is performed without modifying the implementation of the offered cloud services.

SDPA 4.13 Marketing Strategies

4.13.1 Market analysis is performed regularly to identify target audiences and their requirements.

4.13.2 Competitor analysis is performed to develop new market plans.

4.13.3 Providers offer a discounted price to increase sales of the SaaS application.

SDPA 4.14 Innovation

4.14.1 The development team uses innovative ideas successfully for development.

4.14.2 Management supports reactive and proactive innovation measures for SaaS development.

4.14.3 An R&D roadmap is successfully used for SaaS development.

4.14.4 There is a well-developed SaaS business model to capture e value from innovation.

SDPA 4.15 Risk mitigation

4.15.1 Risk assessment is considered mandatory during the pre-production phase of SaaS development.

4.15.2 Risk-related tasks are identified by the management team for each milestone.

4.15.3 Management can perform proactive risk management.

SDPA 4.16 Customer satisfaction

4.16.1 Management can gain customer loyalty.

4.16.2 Customers are satisfied but unwilling to pay more to perceive a high outcome.

SDPA 4.17 Business Collaboration

4.17.1 Management team effectively collaborates with other businesses.

4.17.2 The management team actively performs a market analysis to find opportunities for collaboration.

SDPA 4.18 Competitors

4.18.1 Competitors influence the whole SaaS development process and its features

4.18.2 Market study about specific products offered is important to stay competitive.

4.18.3 Competitor analysis is performed to develop new market plans.

4.18.4 Competitor analysis is performed to decide on a pricing strategy.

SDPA 4.19 Organizational learning

4.19.1 On new technological advancements, training programs are arranged by the organization for employees.

SDPA 4.20 Change management

4.20.1 The plan for any change in the project is well communicated to all project members.

4.20.2 Effective communication during change management helped to prevent resistance to change.

4.20.3 Organizational changes are based on the requirements of the targeted SaaS market segment.

SDPA 4.21 Conflict management

4.21.1 A conflict management policy exists to handle conflicts in project teams.

4.21.2 Management supports positive and constructive conflict.

SDPA 4.22 Organizational structure

4.22.1 The roles and responsibilities of individuals and groups are well-defined and documented in the organization.

4.22.2 A solid and open communication channel among various organization entities is present.

SDPA 4.23 Organizational culture

4.23.1 Collaborative culture in the organization fostered high customer satisfaction.

Level 5: ***Optimized***

This is the highest level of our maturity model and is called ''optimized''. All identified SDPAs play a critical part in the performance of developed SaaS projects. At this level, the organization optimizes its SDPAs, and management and development teams collaborate to develop and manage SaaS projects efficiently. The organization has a domain expert for domain analysis, who has been used to design reusable SaaS components. The development team has the required resources and skills and is seeking knowledge about new technologies for SaaS development. The developing team has developed an optimized fault-tolerant architecture with good performance and reliability, and parallel processing is helping with fast application processing. The organization uses process-based customization tools, Standardized SaaS integration techniques and automated identification of root causes for security alerts. At this level, customers are satisfied and willing to pay more to perceive a high outcome, and sales are steadily increasing. The resulting statements listed below apply to an organization at level 5.

SDPA 5.1 Domain analysis

5.1.1 Domain analysis experts team perform the domain analysis based on guidelines.

5.1.2 Domain analysis has proved to help make reusable SaaS components.

5.1.3 Automated domain analysis is performed for all past SaaS projects.

SDPA 5.2 Design pattern

5.2.1. The hierarchical Multi-Tenant Pattern is used to implement any SaaS application project and identity management systems.

SDPA 5.3 Design for failure

5.3.1 SaaS applications can run smoothly if any server randomly disappears.

5.3.2 Design for failure has proved helpful for designing better applications and supporting fault-tolerant architecture.

SDPA 5.4 Design for Parallelism

5.4.1 Design for parallelism is used to increase the computation power for faster application processing.

SDPA 5.5 QoS differentiation

5.5.1 The well-defined QoS design is used to provide performance differentiation, service granularity, and cost efficiency.

5.5.2 SaaS developers can achieve optimization goals by reducing resource costs and achieving the best system performance.

SDPA 5.6 Customization management

5.6.1 Process-based customization tools are used to facilitate the SaaS provider's fault tolerance and recovery management.

SDPA 5.7 Scalability management

5.7.1 Scalability of multiple levels enables flexibility and control in the SaaS application.

SDPA 5.8 MTA

5.8.1 The evolution of multi-tenant is executed without affecting service continuity.

SDPA 5.9 Security management

5.9.1 Security alerts by monitors and automated identification of the root cause of the alerts using filters and decision trees are working on the SaaS project.

SDPA 5.10 Integration management

5.10.1 The development team provides Integration support in data layers, user interface, and business logic layers.

5.10.2 The development team follows standards to make SaaS integration easy.

5.10.3 Standardized SaaS integration techniques are used to improve integration productivity, efficiency, and SaaS adoption.

SDPA 5.11 Fault tolerance and recovery management

5.11.1 Fault tolerance and recovery management models have a good response time, performance and reliability.

5.11.2 Proactive Fault tolerance techniques are used to predict expected failure and act before the failure happens.

SDPA 5.12 Monitoring management

5.12.1 Monitoring systems can handle different scenarios.

5.12.2 Monitoring tools are scalable to deliver the monitored information in a timely and flexible manner.

5.12.3 Monitoring tools can adapt to a new situation by operating in a changed environment.

SDPA 5.13 Marketing Strategies

5.13.1 The team assesses the value of marketing based on sales success and growth figures.

5.13.2 Marketing strategies are helping to increase the sales of SaaS projects.

SDPA 5.14 Innovation

5.14.1 Past innovative measures taken by the development team have resulted in improved SaaS development and management processes.

SDPA 5.15 Risk mitigation

5.15.1 Risk assessment helps reduce associated development risks.

5.15.2 There is a backup plan to handle identified risks and explore other solutions to reduce or eliminate risk.

5.15.3 Risk mitigation practices have helped the organization to build customer trust in SaaS adoption.

SDPA 5.16 Customer satisfaction

5.16.1 Customers are satisfied and are willing to pay more to perceive a high outcome.

5.16.2 Management can charge a premium price for their product or service.

5.16.3 The sales of SaaS products have steadily increased over the past three years.

5.16.4 The organization can retain existing clients, attract new customers and launch new SaaS applications.

SDPA 5.17 Business Collaboration

5.17.1 Management team successfully collaborates with other businesses while ensuring required security.

5.17.2 The management team is continuously improving its collaboration with the rest of the SaaS market.

SDPA 5.18 Competitors

5.18.1 The organization can gain a competitive advantage using its market orientation strategy.

SDPA 5.19 Organizational learning

5.19.1 The organization learned from its experience and lessons and avoided making mistakes repeatedly.

5.19.2 Organizational learning demonstrated a positive impact on product performance.

SDPA 5.20 Change management

5.20.1 The resistance to change in the organization is gradually decreasing.

SDPA 5.21 Conflict management

5.21.1 The organization has developed on time, reducing costs and application bugs over the past three years.

5.21.2 Cohesion and the effective use of conflict by the top management team facilitated better top management team performance.

5.21.3 The team engaged in functional, task-oriented conflict outperforms those in which conflict is dysfunctional and personally oriented.

SDPA 5.22 Organizational structure

5.22.1 Defined Roles and responsibilities demonstrated a positive impact on the product.

5.22.2 The management team actively revises its defined roles and responsibilities.

SDPA 5.23 Organizational culture

5.23.1 It is not difficult for a new employee to settle down in the organization's working environment.

5.23.2 Team members are satisfied with the current culture of the organization.

D. PERFORMANCE SCALE

The maturity level applies to an organization's process improvement achievement and is determined by its capability to accomplish key SaaS development process areas (SDPAs). The organization's maturity in this Software maturity model can be rated on a five-level Likert scale. A quantitative rating indicates the level of agreement with each statement. The ordinal rating used to measure each dimension's S_{DPAs} is shown in Table 3. This rating consists of five scales: "not applicable", "slightly applicable", "partially applicable", "largely applicable," and finally, "completely applicable". "Not applicable" is added to this rating to increase the flexibility of the method. Although linguistic terms are a little modified according to the design of the maturity model questionnaire, the proposed performance scale and the threshold are structured according to the already validated popular scales, e.g., the BOOTSTRAP methodology. The proposed method enables the organization to assess its SDPAs.

TABLE 3. Performance scale.

Scale	Linguistic expression of proposed performance Scale	Linguistic expression of BOOTSTRAP	Rating threshold (%)
4	Completely applicable	Completely satisfied	≥80
3	Largely applicable	Largely satisfied	66.7 – 79.9
2	Partially applicable	Partially satisfied	33.3 – 66.6
1	Slightly applicable	Absent/ poor	≤ 33.2
0	Not applicable	-	-

Rating Method: As we stated earlier, the rating methodology is adapted from the BOOTSTRAP algorithm [40]. This rating method uses various terms, including development performance rating (DPR_{DPA}), number of applicable statements (NA_{DPA}), passing threshold (PT_{DPA}), and development maturity level (DML). A description of these terms is given below.

$DPRs_{DPA}[x,y]$ is the rating of the *xth* DPA at the *yth* maturity level and can be rated as follows.

$DPR_{DPA}[x, y]$

$= 4$ if the extent of applicability of the statement is 80%

$= 3$ if the extent of applicability of the statement is from 66.7% to 79.9%

$= 2$ if the extent of applicability of the statement is from 33.3% to 66.6%

$= 1$ if the extent of applicability of the statement is less than 33.3%

$= 0$ if the statement is not applicable at all.

If DPR_{DPA} $[x,y] \geq 3$ or DPR_{DPA} $[x,y]$ is equal to 0, then the *xth* statement is considered applicable at the *yth* maturity level. The following expression defines the number of applicable statements NA_{DPA} at the yth maturity level.

$NA_{DPA}[y]$

$= \text{Number of } \{DPR_{DPA}[x, y]|\text{Applicable}\}$

$= \text{Number of } \{DPR_{DPA}[x, y]|DPR_{DPA}[x, y] \geq 3 \text{or } DPR_{DPA}[x, y] = 0\}.$

If 80% of the statements in the corresponding questionnaire are applicable, then the specific maturity level is considered accomplished. The passing threshold for each maturity level is shown in Table 4. Hence, if NADPA [y] is the total number of statements at the *yth* maturity level, then PTDPA at the *yth* maturity level is defined as follows:

$$PT_{DPA}[y] = NA_{DPA}[y] * 80\%.$$

TABLE 4. Rating thresholds.

SaaS maturity level	Total questions	Passing threshold
Ad-hoc	35	28
Opportunistic	39	31
Consistent	38	30
Organized	55	44
Optimized	44	35

The highest maturity level for which the number of applicable statements is equal to or greater than the passing threshold is considered as SaaS maturity level and is defined as:

$$\text{SaaS Maturity level ML} = \max\{y|NA_{DPA}[y] \geq PT_{DPA}[y]\}$$

IV. CASE STUDY

A. ASSESSMENT METHODOLOGY

The proposed SaaS maturity model was applied to two SaaS application development organizations to assess their maturity level. Based on the signed contract, these organizations agreed to participate in this assessment so that their name and individual and organizational identities would not be disclosed. The paper refers to Organizations A and B. Organization A builds various SaaS applications from all genres, such as health, education and others. Organization B is another SaaS application development company that provides advanced SaaS product development services for mobile and web apps. The summarized assessment results for both organizations are presented in Table 5. The questionnaire is designed to assess the maturity profile of the organization. It aims to assess the SaaS process design and development practices, the application's business performance, and best organizational practices. The respondents' responses were captured using the Likert scale ranging from 0 to 4 to indicate the extent of their agreement with each statement in the measuring instrument. A statement is considered applicable if the performance rating is equal to or greater than three or equal to 0 per rating methodology. The responses are depicted in Table 5.

The respondents were either project managers or members of the development team. The survey link or MS teams were used to collect the responses. Participation in the assessment is voluntary without any compensation. The details of both case studies are presented in the subsequent section. Multiple responses were recorded from each organization to avoid bias in the data sample. The respondents provided their observations about the development practices. Further, an inter-rate agreement analysis was performed to avoid any chance of biased responses.

TABLE 5. Responses data for the maturity assessment.

SaaS maturity Level	Total questions	Passing threshold 80%	Organization "A" NA_{DPA}	Organization "B" NA_{DPA}
Ad-hoc	35	28	33	32
Opportunistic	39	31	36	35
Consistent	38	30	30	25
Organized	55	44	40	38
Optimized	44	35	30	25

B. ORGANIZATION "A"

Organization A provides cloud-based solutions for small and mid-size companies (SMEs). The organization has grown globally, offering products such as custom business analytics, project management tools and CRM systems. They use agile development and beta testing. It has around 400-450 clients and is classified as a medium-sized organization. In the study's first phase, personal referrals were used to get consent for the assessment. The study's respondents were selected based on the number of years of service. The individuals were selected based on at least three years of service. The personalized emails were sent to state the study's objective and seek their consent to participate. The one-on-one sessions were also conducted for the Q and A.

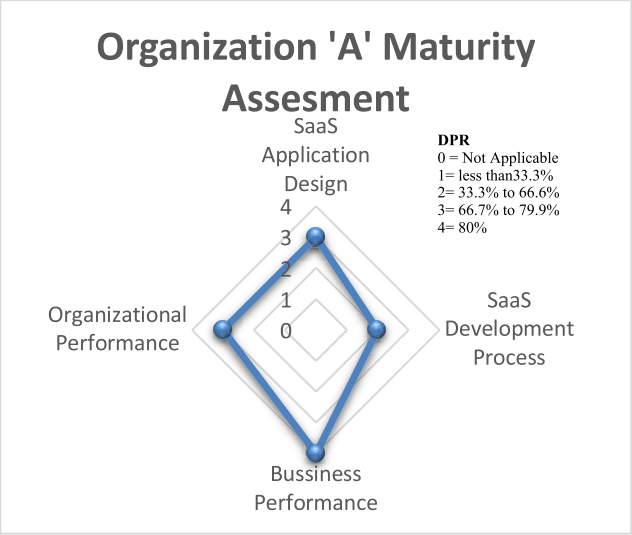

FIGURE 3. Organization 'A' maturity assessment.

The respondents agreed to participate in the study based on the guarantee that their names or specific information would not be disclosed in any subsequent publication. After the first phase, the assessment questionnaires were shared with the participants. They completed the questionnaires for each dimension and level. The questionnaires were designed for each level of maturity of their SaaS application development practices. The respondents use a performance scale ranging from 0 to 4 from Table 3 to provide their extent of agreement with the specific statement. The assessment questions are presented using a bottom-up approach, where the respondents have to start from the level 1 questionnaire and progress to higher levels. The different-level questionnaire was also designed based on the bottom-up approach. The approach follows more advanced features introduced as the respondents move from a lower to a higher level.

The organization "A" respondents had different roles, such as developer, policy-making, and strategic implementation.

Data Analysis: Once the survey was complete, each level's NA_{DPA} (number of applicable statements) was calculated. NA_{DPA} was 33 for Level 1, 36 for Level 2, 30 for Level 3, 40 for Level 4, and 30 for Level 5. Therefore, Organization "A" passed the rating threshold of 80% for Level 3, and consequently, Organization "A" is at the "Consistent" level.

For further analysis, we used the radar charts for each dimension of the maturity model based on the rating methodology. Figure 4 demonstrates the average responses for each dimension at level 3. The chart shows four dimensions of a SaaS maturity model, demonstrating DPR ranging from 0 to 4, and it depicts the respondents' agreement level

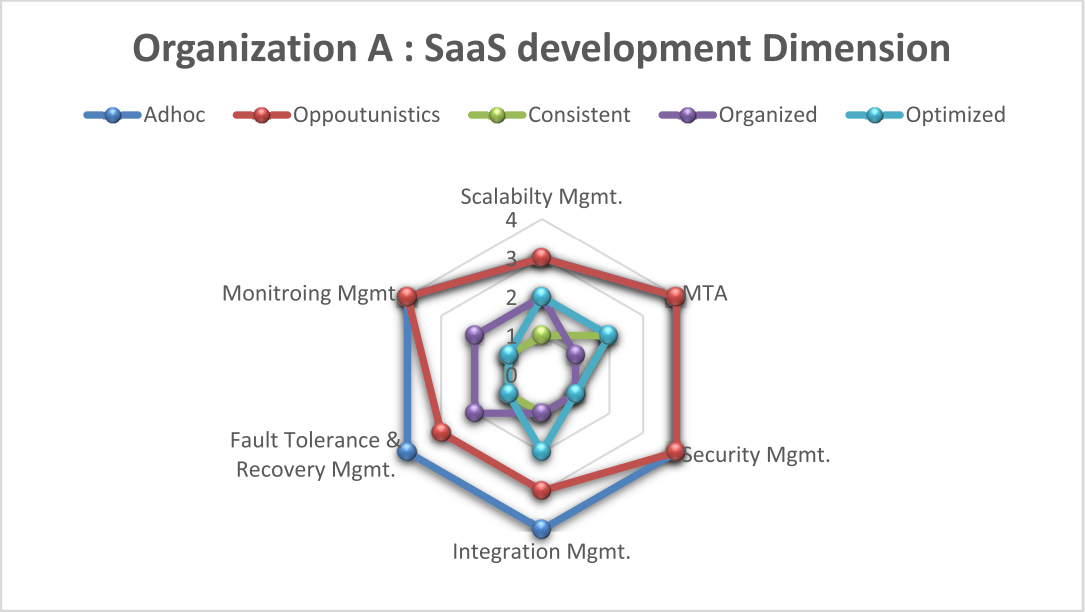

FIGURE 4. Organization 'A' SaaS development dimension results.

with the statement. Fig. 3 clearly shows that organization 'A' needs to improve its SaaS development dimensions practices, specifically, and must look into game design strategy dimensions and business performance practices. This analysis helps the organization examine their practices for the specific domains. Fig. 4 shows the spiral chart for the SaaS development dimension based on the respondents' average responses. It clearly shows that the organization needs to improve its fault-tolerance management.

This will also help them identify their gaps and understand how to achieve higher levels. To dig further into each dimension, for example, Fig. 4 depicts the SaaS development process spiral chart for all levels of the SaaS maturity model based on the respondents' responses average.

It shows that under the SaaS development dimension, organization 'A' needs to improve its fault tolerance management, MTA, monitoring management, and security practices to move from level 2 to 3. Moreover, they must improve their overall development processes dimension practices to achieve a higher level.

C. ORGANIZATION "B"

The second participation organization is Organization "B", another small SaaS application development organization. It focuses on developing innovative solutions for large companies and customizable analytics. Their development approach is microservices architecture, which is continuously deployed. A similar approach is used as organization A to establish the contact. The respondents of Organization B have been working in the company for four years. The respondents of this organization also used the same performance scale to indicate their extent of agreement with the statements, and they responded to the questionnaire about each dimension and level. The respondents had roles of software engineers, data scientists, and business analysts.

Data Analysis: After completing the survey, each level's NA_{DPA} (number of applicable statements) was again calculated. NA_{DPA} was 32 for Level 1, 35 for Level 2, 25 for Level 3, 38 for Level 4, and 25 for Level 5. Therefore, Organization "B" passed the rating threshold of 80% for Level 2, and consequently, Organization "B" is at the "Opportunistic" level. A similar method is used for data analysis for organization B. Fig. 5 shows the response averages for each dimension at level 2. The chart depicts four dimensions of a SaaS maturity model: the rating method and the performance scale. It indicates that Organization "B" needs to improve its SaaS development practices to achieve higher levels, as most respondents selected DPR 2 at other levels. Furthermore, they need to improve their practices for other dimensions. Fig 6. depicts the SaaS development dimension results for all levels.

D. INTER-RATER AGREEMENT ANALYSIS

There is always a chance of a conflict of opinion about SaaS development practices in the organization when there is more than one respondent from that organization. Inter-rater agreement analysis [58] was performed to solve this issue. It is a popular method to calculate the extent of agreement in the ratings from different participants for the same process or software engineering practice [59]. In this study, inter-rater agreement analysis was performed to calculate the extent of agreement among respondents from the same SaaS organization. Kendall's W, also known as Kendall coefficient of concordance (W), [60] is a preferred non-parametric statistic for ordinal data to evaluate inter-rater agreement. "W" represents the difference between the actual agreement as drawn from the data and the perfect agreement. Fleiss'

TABLE 6. Inter-rater agreement analysis of Organizations "A" and "B".

SaaS maturity level	Organization A				Organization B			
	Kendall's Coefficient of Concordance		Fleiss Kappa statistics		Kendall's Coefficient of Concordance		Fleiss Kappa statistics	
	Coef.(W)	X2	Coef.	Z	Coef.(W)	X2	Coef.	Z
Ad-hoc	0.63	50.91*	0.63	7.01*	0.87	52.24*	0.63	4.79*
Opportunistic	0.63	52.10*	0.62	7.66*	0.91	47.66*	0.76	5.19*
Consistent	0.72	58.13*	0.66	7.98*	0.85	54.61*	0.74	4.34*
Organized	0.70	57.10*	0.67	8.01*	0.69	54.32*	0.65	5.01*
Optimized	0.72	54.74**	0.69	7.91**	0.85	52.44**	0.80	5.13*

Significant at $p<0.01$*; Significant at $p<0.05$**.

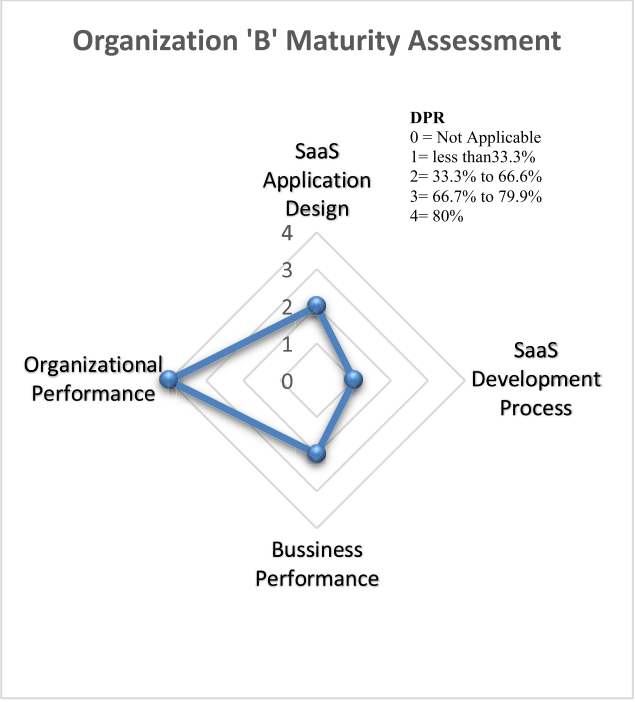

FIGURE 5. Organization 'B' maturity assessment.

kappa is another statistical measure of agreement between three or more raters. It is widely used when a researcher wants to determine how well raters agree on coding nominal variables [61].

The value of Fleiss Kappa and Kendall's *W can range* from 0 (representing complete disagreement) to 1 (representing perfect agreement) [61]. The Kappa [59] standard consists of four levels: > 0.78 represents excellent agreement, 0.62 to 0.78 indicates substantial agreement, 0.44 to 0.62 represents moderate agreement, and < 0.44 means poor agreement. In this study, the observed Kappa coefficient fell into the substantial category, ranging from 0.65 to 0.69. Table 6 reports both organizations' Kappa and Kendall statistics; both measures fall into the "substantial agreement" category.

V. DISCUSSION

The maturity model helps to obtain complete insight into current development processes, their related activities, and the current level of maturity for a software engineering organization. It provides a structured way for organizations to approach problems and challenges by providing a benchmark against which they can assess their capabilities and a roadmap for improving them. The maturity model motivates us to adopt best practices and struggles to the next level.

SaaS development process and management are very competitive and challenging areas for research. The SaaS architecture design, database partitioning, database architecture, scalability issues, user interface design, use of APIs and workflow are different from traditional applications, which introduces many challenges during the design and operation of a SaaS system. Furthermore, SaaS vendors develop and maintain the system on behalf of their customers, which requires a dedicated platform and continuous monitoring to avoid any service barrier. Unique requirements from different customers may increase the operating cost. In addition, the complexity of multi-tenant architecture can lead to poor performance and low resource utilization. In this highly competitive and challenging SaaS market, determining the maturity of a current process or a specific area in the SaaS development process that needs improvement is critical. Due to the rapidly evolving nature of the domain, the topic of development strategies and best practices for SaaS development has not been fully explored. The authors conducted empirical investigations and a literature review to identify the key factors and their impact on SaaS development methodology. Research models have been developed based on an examination of key factors. To assess SaaS development practices, the significant key factors identified from four empirical investigations have been used as a measuring instrument to develop a SaaS maturity model. The structure of the SaaS maturity model is composed of a four-dimensional assessment framework. The SaaS maturity model has been used to measure the maturity level of development practices using the developed assessment methodology and conducting case studies. SaaS application development organizations can use the model proposed in this paper to assess and improve their current practices. It also helps organizations to notice bottlenecks in their existing policies and methodologies. Consequently, it results in the development of successful SaaS products, directly influencing the overall business dimension of SaaS organizations. This assessment and evaluation help

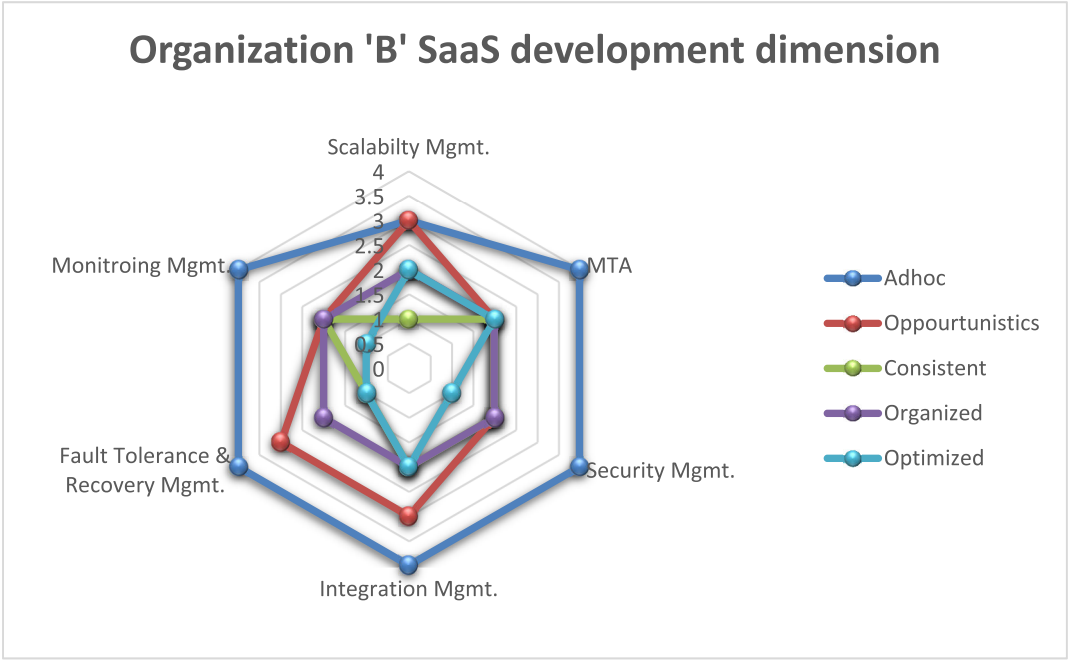

FIGURE 6. Organization 'B' SaaS development dimension results.

the development team to recognize the key process areas of SaaS development.

A. LIMITATIONS OF THE STUDY

Level-based questionnaires are used for assessment in the proposed SaaS maturity model. Thus, it could be prone to some limitations. Although twenty-three key factors are highlighted for five levels of maturity based on four empirical studies, it is always possible that other factors, such as organization size and cultural and economic conditions, have been inadvertently excluded. The proposed methodology is based on the responses from team members and managers. However, approaches used to ensure reliability and validity are part of the common statistical techniques used by software engineering researchers. Typically, independent assessors are considered essential when defining coordination with the internal assessment team. Still, in the proposed methodology, we did not consider the role of an independent assessor, and case studies are performed based on self-assessment.

The proposed model provides numerical data regarding the maturity level of organizational factors and SaaS development practices, but no guidelines are provided to improve them. Our future research will include guidelines on improving from lower to higher levels. While the presented model has some specific and general limitations, commonly used statistical approaches have been used to validate key SaaS development process areas. It is a comprehensive method to get insights into current SaaS development practices. This study also provides future directions for research in SaaS development practices.

VI. CONCLUSION

In the last few years, Software as a Service (SaaS) has proved to be one of the most promising service delivery models in cloud computing. SaaS providers offer nearly all application functional extensions and add-ons as a service, and SaaS solutions deliver better business outcomes than traditional software.

The SaaS delivery model has multiple advantages for the consumer and the provider. It has different architecture requirements, database design, user interface design, and use of APIs and workflow than traditional software applications. Consumers are also concerned about data privacy, security, availability, and SLA. To fulfill the unique SaaS development requirement and address the customer concerns, there is a compelling need for a method to assess the current practices in the organization.

Assessment of SaaS development processes is an important area of research for developing quality SaaS to improve the organization's position in the current competitive SaaS market. To our knowledge, no comprehensive research has been reported regarding SaaS development methodology assessment that considers all key factors of SaaS development and management. This paper has proposed a SaaS maturity model that includes key SaaS development factors and vital software engineering and project management concepts. To construct the model framework, SaaS development and management have been divided into four dimensions: design, architecture, organization, and business performance. This can be applied to an organization of any size and domain. The framework includes assessment questionnaires for the five maturity levels, a rating method, and a performance scale. Case studies

were conducted on two SaaS organizations to demonstrate and validate the proposed model. This model proposed the best practices for managing complex SaaS projects. It also helps managers to find the areas that require improvement.

REFERENCES

[1] S. Satyanarayana, "Cloud computing: SAAS," *Computer Sciences Telecommunications*, vol. 4, pp. 76–79, Jun. 2012.
[2] T. Koraza. (2024). *Software as a Service (SaaS) Market Growth and Trends in 2024*. Accessed: Apr. 30, 2024. [Online]. Available: https://www.madx.digital/learn/saas-market
[3] C.-P. Bezemer and A. Zaidman, "Multi-tenant SaaS applications: Maintenance dream or nightmare?" in *Proc. Joint ERCIM Workshop Softw. Evol. (EVOL) Int. Workshop Princ. Softw. Evol. (IWPSE)*, Sep. 2010, pp. 88–92.
[4] J. Ju, Y. Wang, J. Fu, J. Wu, and Z. Lin, "Research on key technology in SaaS," in *Proc. Int. Conf. Intell. Comput. Cognit. Informat.*, Jun. 2010, pp. 384–387.
[5] Y. Cao, C.-H. Lung, and S. A. Ajila, "Constraint-based multi-tenant SaaS deployment using feature modeling and XML filtering techniques," in *Proc. IEEE 39th Annu. Comput. Softw. Appl. Conf.*, vol. 3, Jul. 2015, pp. 454–459.
[6] W. Su, J. Hu, C. Lin, and S. Shen, "SLA-aware tenant placement and dynamic resource provision in SaaS," in *Proc. IEEE Int. Conf. Web Services*, Jun. 2015, pp. 615–622.
[7] Y.-H. Wang, "The role of SaaS privacy and security compliance for continued SaaS use," in *Proc. 7th Int. Conf. Networked Comput. Adv. Inf. Manage.*, Jun. 2011, pp. 303–306.
[8] J. Y. Lee, J. W. Lee, D. W. Cheun, and S. D. Kim, "A quality model for evaluating software-as-a-service in cloud computing," in *Proc. 7th ACIS Int. Conf. Softw. Eng. Res., Manage. Appl.*, Dec. 2009, pp. 261–266.
[9] C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, and B. Gao, "A framework for native multi-tenancy application development and management," in *Proc. 9th IEEE Int. Conf. e-commerce Technol. 4th IEEE Int. Conf. Enterprise Comput.*, Jul. 2007, pp. 551–558.
[10] W.-T. Tsai, Y. Huang, and Q. Shao, "EasySaaS: A SaaS development framework," in *Proc. IEEE Int. Conf. Service-Oriented Comput. Appl. (SOCA)*, Dec. 2011, pp. 1–4.
[11] J. Lee, S. Kang, and S. Jin Hur, "Web-based development framework for customizing java-based business logic of SaaS application," in *Proc. 14th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2012, pp. 1310–1313.
[12] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, "Capability maturity model, version 1.1," *IEEE Softw.*, vol. 10, no. 4, pp. 18–27, Jul. 1993.
[13] A. April, J. H. Hayes, A. Abran, and R. Dumke, *Softwaremaintenance Maturity Model*. Hoboken, NJ, USA: Wiley, 2004.
[14] S. Aleem, F. Ahmed, R. Batool, and A. Khattak, "Empirical investigation of key factors for SaaS architecture," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1037–1049, Jul. 2021, doi: 10.1109/TCC.2019.2906299.
[15] S. Aleem, R. Batool, F. Ahmed, A. Khatak, and R. M. U. Ullah, "Architecture guidelines for SaaS development process," in *Proc. Int. Conf. Cloud Big Data Comput.*, Sep. 2017, pp. 94–99, doi: 10.1145/3141128.3141136.
[16] S. Aleem, R. Batool, F. Ahmed, and A. Khattak, "Empirical invesitigation for business factors for SaaS organization," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1037–1049, Mar. 2024.
[17] S. Aleem, R. Batool, F. Ahmed, and A. Khattak, "Guidelines for key organizational factors for SaaS organizations," in *Proc. 3rd Int. Conf. Big Data Res.*, 2020, pp. 83–87.
[18] R. Shvydun. (2024). *Future of SaaS Trends and Predictions*. Accessed: Apr. 30, 2024. [Online]. Available: https://www.custify.com/blog/future-of-saas-trends-and-predictions-2024/
[19] M. Fan, S. Kumar, and A. B. Whinston, "Short-term and long-term competition between providers of shrink-wrap software and software as a service," *Eur. J. Oper. Res.*, vol. 196, no. 2, pp. 661–671, Jul. 2009.
[20] S. Walraven, E. Truyen, and W. Joosen, "Comparing PaaS offerings in light of SaaS development," *Computing*, vol. 96, no. 8, pp. 669–724, Aug. 2014.
[21] D. Jagli, S. Purohit, and N. S. Chandra, "SAASQUAL: A quality model for evaluating SaaS on the cloud computing environment," in *Advances in Intelligent Systems and Computing*. Cham, Switzerland: Springer, 2018, pp. 429–437.
[22] H. La and S. Kim, "A systematic process for developing high quality SaaS cloud services," *Cloud Comput.*, vol. 1, pp. 278–289, May 2009.
[23] J. Espadas, D. Concha, and A. Molina, "Application development over software-as-a-service platforms," in *Proc. 3rd Int. Conf. Softw. Eng. Adv.*, Oct. 2008, pp. 97–104.
[24] A. K. Mandal, S. Changder, A. Sarkar, and N. C. Debnath, "Architecting software as a service for data centric cloud applications," *Int. J. Grid High Perform. Comput.*, vol. 6, no. 1, pp. 77–92, Jan. 2014.
[25] J. Espadas, D. Concha, D. Romero, and A. Molina, "Open architecture for developing multitenant software-as-a-service applications," in *Proc. 1st Int. Conf. Cloud Comput.*, 2010, pp. 92–97.
[26] R. Mietzner, F. Leymann, and M. P. Papazoglou, "Defining composite configurable SaaS application packages using SCA, variability descriptors and multi-tenancy patterns," in *Proc. 3rd Int. Conf. Internet Web Appl. Services*, Jun. 2008, pp. 156–161.
[27] S. Kang, S. Kang, and S. Hur, "A design of the conceptual architecture for a multitenant SaaS application platform," in *Proc. 1st ACIS/JNU Int. Conf. Comput., Netw., Syst. Ind. Eng.*, May 2011, pp. 462–467.
[28] R. Mathew and R. Spraetz, "Test automation on a SaaS platform," in *Proc. Int. Conf. Softw. Test. Verification Validation*, Apr. 2009, pp. 317–325.
[29] G. Liu, "Research on independent SaaS platform," in *Proc. 2nd IEEE Int. Conf. Inf. Manage. Eng.*, Apr. 2010, pp. 110–113.
[30] H. He, "Applications deployment on the SaaS platform," in *Proc. 5th Int. Conf. Pervasive Comput. Appl.*, Dec. 2010, pp. 232–237.
[31] Z. Jiang, W. Sun, K. Tang, J. L. Snowdon, and X. Zhang, "A pattern-based design approach for subscription management of software as a service," in *Proc. Congr. Services I*, Jul. 2009, pp. 678–685.
[32] K. Tang, J. M. Zhang, and Z. B. Jiang, "Framework for SaaS management platform," in *Proc. IEEE 7th Int. Conf. E-Business Eng.*, Nov. 2010, pp. 345–350.
[33] D. Dempsey and F. Kelliher, "Business-to-business client relationships in the cloud computing software as a service realm," in *Industry Trends in Cloud Computing*. Cham, Switzerland: Springer, 2018, pp. 83–109.
[34] D. Candeia, R. A. Santos, and R. Lopes, "Business-driven long-term capacity planning for SaaS applications," *IEEE Trans. Cloud Comput.*, vol. 3, no. 3, pp. 290–303, Jul. 2015.
[35] J. P. Lawler and H. Howell-Barber, "A framework model for a software-as-a-service (SaaS) strategy," in *Advances in Information Quality and Management*. Hershey, PA, USA: IGI Global, 2014, pp. 1024–1032.
[36] Z. Pervez, S. Lee, and Y.-K. Lee, "Multi-tenant, secure, load disseminated SaaS architecture," in *Proc. 12th Int. Conf. Adv. Commun. Technol. (ICACT)*, vol. 1, Feb. 2010, pp. 214–219.
[37] G. Laatikainen and A. Ojala, "SaaS architecture and pricing models," in *Proc. IEEE Int. Conf. Services Comput.*, Jun. 2014, pp. 597–604.
[38] A. Fuggetta, "Software process: A roadmap," in *Proc. Conf. Future Softw. Eng.*, May 2000, pp. 1–12.
[39] M. Paulk, "Capability maturity model for software," *Encycl. Softw. Eng.*, vol. 1, no. 1, pp. 1–24, 1993.
[40] C. P. Team. (2006). *CMMI for Development*. [Online]. Available: https://insights.sei.cmu.edu/library/cmmi-for-development-version-12/
[41] P. Kuvaja, "BOOTSTRAP: A software process assessment and improvement methodology," in *Objective Software Quality*. Cham, Switzerland: Springer, 1995, pp. 31–48.
[42] A. Dorling, "SPICE: Software process improvement and capability determination," *Softw. Quality J.*, vol. 2, no. 4, pp. 209–224, Dec. 1993.
[43] M. Pyhajarvi. (2004). *SPICE: International Standard for Software Process Assessment*. [Online]. Available: http://www.cs.helsinki.fi/u/paakki/Pyhajarvi.pdf
[44] M. J. Parzinger, R. Nath, and M. A. Lemons, "Examining the effect of the transformational leader on software quality," *Softw. Quality J.*, vol. 9, no. 4, pp. 253–267, 2001.
[45] R. Singh, "International standard ISO/IEC 12207 software life cycle processes," *Softw. Process, Improvement Pract.*, vol. 2, no. 1, pp. 35–50, Mar. 1996.
[46] F. Chong and G. Carraro, "Architecture strategies for catching the long tail," *MSDN Libr. Microsoft Corp.*, vol. 1, pp. 9–10, Jun. 2006.
[47] S. Ried, J. R. Rymer, and R. Iqbal, "Forrester's SaaS maturity model: Transforming vendor strategy while managing customer expectations," *Forrester Res.*, vol. 1, pp. 1–29, Aug. 2008.
[48] S. Kang, "A general maturity model and reference architecture for SaaS service," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2010, pp. 337–346.
[49] X. Chen, "A service quality based evaluation model for SaaS systems," 2010.

[50] B. Wang, H. Liu, and J. Song, "SaaS-based enterprise application integration approach and case study," *J. Supercomput.*, vol. 72, no. 7, pp. 2833–2847, Jul. 2016.
[51] A. V. Hudli, B. Shivaradhya, and R. V. Hudli, "Level-4 SaaS applications for healthcare industry," in *Proc. 2nd Bengaluru Annu. Compute Conf.*, Jan. 2009, p. 19.
[52] W. Li, Z. Zhang, S. Wu, and Z. Wu, "An implementation of the SaaS level-3 maturity model for an educational credit bank information system," in *Proc. Int. Conf. Service Sci.*, May 2010, pp. 283–287.
[53] M. H. Cancian, R. J. Rabelo, and J. C. R. Hauck, "Towards a capability and maturity model for collaborative software-as-a-service," *Innov. Syst. Softw. Eng.*, vol. 16, nos. 3–4, pp. 245–261, Dec. 2020.
[54] W. Gunawan and E. Setyawan, "Applying effective cloud computing maturity model (CCMM)," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 725, no. 1, pp. 1–16, 2020.
[55] V. Hegde, "Reliability in the medical device industry," *Handb. Performability Eng.*, vol. 1, pp. 997–1009, May 2008.
[56] E. K. Clemons and Y. Chen, "Making the decision to contract for cloud services: Managing the risk of an extreme form of IT outsourcing," in *Proc. 44th Hawaii Int. Conf. Syst. Sci.*, Jan. 2011, pp. 1–10.
[57] L. V. Manzoni and R. T. Price, "Identifying extensions required by rup (rational unified process) to comply with CMM (capability maturity model) levels 2 and 3," *IEEE Trans. Softw. Eng.*, vol. 29, no. 2, pp. 181–192, Feb. 2003.
[58] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, p. 159, Mar. 1977.
[59] K. El Emam, "Benchmarking Kappa: Interrater agreement in software process assessments," *Empir. Softw. Eng.*, vol. 4, no. 2, pp. 113–133, 1999.
[60] P. K. Choudhary, *Analyzing Rater Agreement: Manifest Variable Methods*. New York, NY, USA: Taylor & Francis, 2006.
[61] J. J. Randolph. (2005). *Free-Marginal Multirater Kappa (Multirater K [Free]): An Alternative To Fleiss' Fixed-Marginal Multirater Kappa*. [Online]. Available: https://eric.ed.gov/?id=ED490661

SAIQA ALEEM (Senior Member, IEEE) received the M.S. degree in computer science from the University of Central Punjab, Pakistan, in 2004, the M.S. degree in information technology from United Arab Emirates University (UAEU), in 2013, and the Ph.D. degree in electrical and computer engineering from the University of Western Ontario, Canada, in 2016. Currently, she is an Assistant Professor with Wilfrid Laurier University. She had many years of academic and industrial experience, holding various technical positions. She has authored and co-authored many peer-reviewed research articles on machine learning techniques applications, software development assessment processes, and software product lines.

RABIA BATOOL received the B.S. degree in information technology from the National University of Science and Technology, Pakistan, and the M.S. degree in biomedical engineering major in computer science from Kyung Hee University, South Korea. After the master's degree, she joined Millennium Software, Pakistan. Currently, she is a Research Assistant with Zayed University. She has worked on many natural language processing projects. During her education, she was awarded Outreach and the Global IT Scholarship.

SHAYMA ALKOBAISI received the Ph.D. degree in computer science from the University of Denver, in June 2008. In 2010, she was appointed the Manager of that program for one year. She also held the position of the Acting Dean of the University College, UAEU, from 2021 to 2022. She is currently an Associate Professor with the College of Information Technology (CIT), United Arab Emirates University (UAEU). She served in different admin positions, including the Department Chair, the Acting Dean, and the Vice Dean of CIT for several years. Recently, she has been working on the field of exposome, the effect of environmental factors on health, using environmental sensors and mobile technology to collect the data, and using GIS, and data mining tools to analyze the data. She has more than 40 peer-reviewed articles published in international journals and conference proceedings, such as *Journal of GeoInformatica*, *Journal of Spatial Information Science*, *Journal of Transactions on Information and Systems*, and *Advances in Water Resources* journal. Her research interests include uncertainty management in spatiotemporal databases, online query processing in spatial databases, geographic information systems, and health informatics. In 2009, she was selected as a member of the Advisory Council to the Science and Technology Program at the Emirates Foundation for Philanthropy, Abu Dhabi. She is also a recipient of several local and international grants.

FAHEEM AHMED received the M.S. and Ph.D. degrees in software engineering from Western University, Ontario, Canada, in 2004 and 2006, respectively. Currently, he is a Full Professor with Thompson Rivers University, Kamloops, Canada. He had many years of industrial experience, holding various technical positions in software development organizations. During his professional career, he has been actively involved in the life cycle of the software development process including requirements management, system analysis and design, software development, testing, delivery, and maintenance. He has authored and co-authored many peer-reviewed research papers in leading journals and conference proceedings in the area of software engineering.

ASAD MASOOD KHATTAK (Senior Member, IEEE) is currently a Graduate Program Coordinator with the College of Technological Innovation, Zayed University. He has more than 17 years of industry and academic experience in various capacities, such as a Webmaster, a Developer, a Team Leader, a Principal Investigator, the Assistant Chair, a Graduate Program Coordinator, an Assistant, an Associate, and a Full Professor. He has multicultural teaching experience and has taught in Pakistan, China, South Korea, and United Arab Emirates (UAE). He has experience of leading research teams of more than 40 members in more than four countries. He has served as the Assistant Chair. He is open to explore more and learn more that has mutual benefits for him, his partners, and his institution. He has delivered various keynote speeches, invited talks, invited lectures, and short courses. He has authored more than 170 publications (books, book chapters, journal publications, and conference publications). He has U.S. patents registered and Korean patents. His research interests include data curation and representation, data mining, semantic web, machine learning, deep learning, computational intelligence, e and m-healthcare, smart cities, and secure computing. Taking bold, decisive, and definitive actions to solve problems, building strategic alliances, and working in collaborative teams are his strengths. He was appointed as an ACM Distinguish Speaker.