A Framework of Service Level Agreement Management for Software Maintenance

Syed Irtiza Hassan[1], Ahmad Salman Khan[2]
[1]Department of Computer Science & IT, The University of Lahore, Lahore, Pakistan
[2]Department of Software Engineering, The University of Lahore, Lahore, Pakistan

Abstract
Software maintenance is the largest phase of the software development lifecycle. It's a complex activity due to the diverse nature of services provided during this phase. Therefore, it is significant that the parties involved in maintenance should decide and agree upon the nature and level of maintenance services upfront. This paper discusses the important issue of Service Level Agreement for software maintenance. We conducted a systematic literature review to explore the state of the art and then applied the grounded theory method for formulating the preliminary SLA management framework. We elicited six major phases of SLA management including Service Templates Development, SLA Negotiation, Service Deployment, Service Execution, Service Assessment, and Service Decommissioning. This paper presents a framework for comprehensively managing software maintenance service level agreement. Case studies were conducted in five IT organizations that perform software maintenance to validate and enhance the framework. Both quantitative and qualitative analysis was performed and the refined framework reflected the best practices from industry enhancements were made in every phase of the framework. Furthermore, it was identified that there is a gap in industrial practice as far as the process of service deployment, assessment and corrective actions over SLA and adjustment patterns and manageability for modifying SLA are concerned.

Key words:
service level agreement, software maintenance

1. Introduction

The primary focus of software development is the satisfaction of user requirements. Software products inherently go through change and evolution. Once operational, software product enters a maintenance phase. Maintenance phase comprises of identifying and rectifying defects, an adaptation of system in case of a variation in the operating environment, and the addition of new features based on new user requirements. Although the maintenance is mainly concerned with the support delivered after implementation, maintenance activities start quite earlier parallel to development. SLA for software maintenance is an important area of research; however, little work has been done on SLA with the perspective of software maintenance. General SLA exists but there exists no comprehensive framework specific to software maintenance. The preliminary framework has been published as "Eliciting Theory for Software Maintenance SLA Management Framework", by the authors in December 2017 in International Conference on Frontiers of Information Technology (FIT) (pp. 241-246). IEEE [9]

In this paper, we will be exploring what is the significance of service level agreements for organizations performing software maintenance, whether they follow any framework to manage their SLA. Furthermore, we have to study how service level agreements are developed in such organizations in the first place, are there any service templates they use. How do negotiations affect the SLA? How are service provisioning and deployment carried out? Is there any process defined for assessment and corrective actions during execution? How is the service actually executed? Is termination and decommissioning of service properly addressed in the SLA? As a result, we will be presenting a refined framework of service level agreement management for Software Maintenance.

2. Research Methodology

The research was conducted in four phases: (1) systematic literature review, (2) eliciting a preliminary framework, (3) exploring the framework in industry and (4) updating the framework. Phase 1 and 2 are previously published [9]. This paper presents phase 3 and 4. Figure 1 illustrates the phases of research

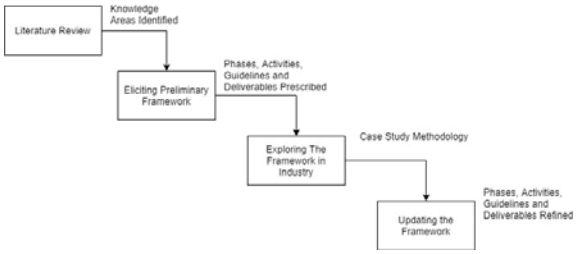

Fig. 1 Research Roadmap

3. Systematic Literature review

We identified the specifications of service Agreements, SLA Requirements, life cycle, modules, negotiation

Manuscript received May 5, 2019
Manuscript revised May 20, 2019

framework along with roles in SLA management and activities unique to software maintenance SLA management through literature updated till the year 2019. An account of the work of the authors is given below.

A. Service level agreement specifications

Service level agreement specifications were discussed by Trienekens, J. J., Bouman, J. J., & Van Der Zwan, M. in 2004 [25]. A structured approach was presented in order to speed up the process of providing service level agreement specifications and to improve the effectiveness of SLAs for both customers and users.

B. Service level agreement activities

Chapin, Hale, Khan, Ramil, and Tan proposed a new classification of software maintenance management activities in 2001. Proactive Software maintenance activities to make the service management process cost-effective and more functional were presented by Lee, Y. J., & Choi, Y. J. in 2015 [15]. An understanding of activities in each software maintenance phase of enterprise systems was presented by Li, S. H., Yen, D. C., Lu, W. H., & Chen, T. Y. in 2014 [17].

C. Roles in SLA management

Kajko-Mattsson, M., Ahnlund, C., & Lundberg, E. presented CM3 a framework for service level agreement in 2004 in which the authors presented roles on SLA management [15].

D. Processes in SLA management

Kajko Mattsson has further presented the processes in SLA Management. EM3: SLA Management process model consists of four main processes in 2009. [13]

E. Unique Software Maintenance activities

April, A., & Abran, A. introduced SMmm, Software maintenance maturity model in 2009 that was inspired by CMMi, it mentioned activities unique to software maintenance that should be included in SLA. [1]

F. Software Maintenance process improvement

Jansson, A. S. provided Software Maintenance and Process Improvement for CMMI in 2007, based on the work of April, Huffman Hayes in. 2005. This work was further improved in 2009 [12,1]
Factors affecting software maintenance work efficiency were presented by Tsunoda, M., Monden, A., Matsumoto, K., Ohiwa, S., & Oshino, T in 2015 [14]. Factors, directly and indirectly, affecting the maintenance process and customer satisfaction was discussed by Christa, S., Madhusudhan, V., Suma, V., & Rao, J. J. in 2017[6]

G. SLA Negotiation

Hasselmeyer, P., Mersch, H., Koller, B., Quyen, H. N., Schubert, L., & Wieder, P. presented a negotiation framework for SLAs in 2007 [10].
A software maintenance requirement negotiation model has been presented by Christa, S., Madhusudhan, V., Suma, V., & Rao, J. J. in [6] and Renegotiation process has been discussed by Huang, H., Hu, M., Kauffman, R., & Xu, H. in 2019 [11]

H. SLA Life Cycle

Life Cycle of SLA was presented by Zhang, S., & Song, M. in 2010 [28]. The study of Kung, H. J., & Hsu, C. developed and classified this concept and introduced the model of the SMLC, which classifies software maintenance into four different stages [29].

I. SLA Modules

An SLA Management Framework has also been prescribed by Comuzzi, M., Kotsokalis, C., Rathfelder, C., Theilmann, W., Winkler, U., & Zacco, G in 2009. Modules of SLA were also defined further [7]

J. Software maintenance service deployment

A software handover framework was presented which is effective for the transition from developer to maintainer by Khan, A. S. in 2013 [16]. Classification and analysis of literature were done in order to scope the phenomenon of the continuous evolution of software functionality was done by Rodríguez, P., Haghighatkhah, A., Lwakatare, L. E., Teppola, S., Suomalainen, T., Eskeli, J., ... & Oivo, M. who mentioned that the software-intensive industry is moving towards the adoption of a value-driven and adaptive real-time business paradigm Software Maintenance Help Desk [24]
A systematic mapping study on software change request repositories was done by Cavalcanti, Y. C., da Mota Silveira Neto, P. A., Machado, I. D. C., Vale, T. F., de Almeida, E. S., & Meira, S. R. D. L in 2014[4]. Mikkonen, T., Lassenius, C., Männistö, T., Oivo, M., & Järvinen, J proposed a software maintenance evaluation method based on users' feedback, by means of the records stored in the help desk's databases system in 2018 [21].

K. Software Maintenance Help Desk

The work of Cook, S. in 2017 explores the measurement of service effectiveness with a focus on customer satisfaction. Chatzimparmpas, A., & Bibi, S explored

factors relevant to the maintainability of a project in 2019 with an objective to improve software service management propose the combination of two well-known machine learning (ML) techniques, Bayesian Networks (BNs), and Association Rules (ARs) for modeling the maintenance process by identifying the relationships among the internal and external quality metrics related to a particular project release to both the maintainability of the project and the maintenance process indicators (i.e., effort and duration) [5].

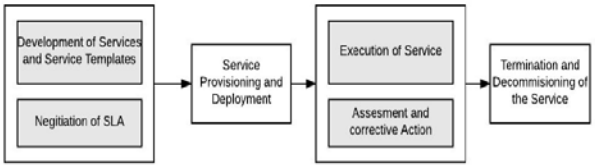

Fig. 2 represents knowledge areas from the systematic literature review along with the authors who contributed to it. This systematic literature review is updated until 2019.

This systemaic literature review gave the basis of applying grounded theory, which is a systematic process to prescribe a preliminary framework. It commences with data collection followed by open coding after which axial or selective coding is done followed by theoretical coding. First off all the available information is coded in order to bring forward concepts out of it. Concepts are than categorized and a collection of these concepts is utilized to formulate a theory.

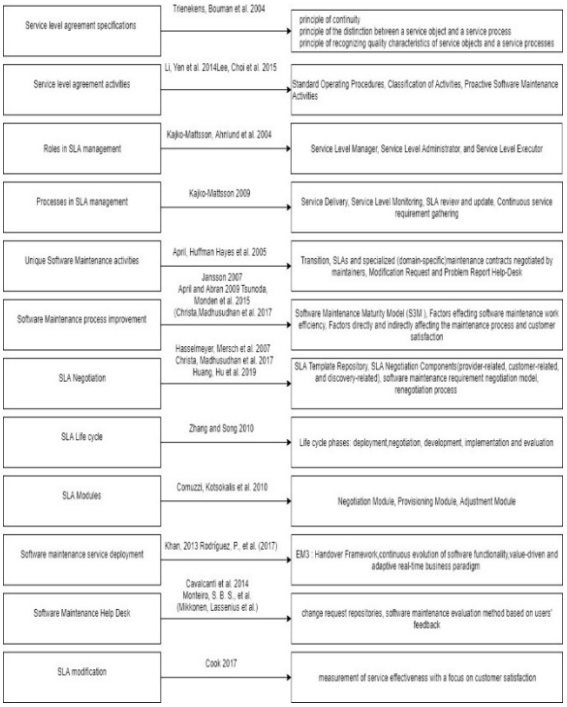

Fig. 3 Systematic literature review knowledge areas

4. Preliminary FrameWork

We elicited six major phases of SLA management including Service Templates Development, SLA Negotiation, Service Deployment, Service Execution, Service Assessment, and Service Decommissioning. This study provides a basis for producing a comprehensive software maintenance SLA to be applicable in an industrial setting. Figure 3 presents the above-mentioned phases. Phases enclosed together in a box represent parallel execution and arrows represent the sequence of phases

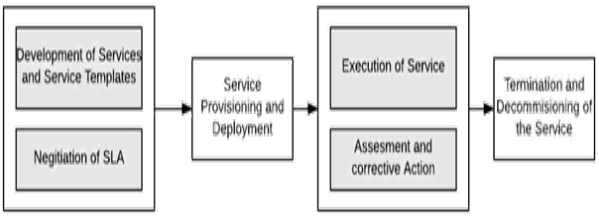

Fig. 4 Preliminary software maintenance SLA management framework

5. Evaluation/Exploration of framework

We applied case study methodology to validate the SLA management Framework for software maintenance. The SLA management Framework for software maintenance has been prescribed through literature. The purpose of this case study was to explore this framework in the industry. To observe as to what extent the phases mentioned in the preliminary framework along with the activities and deliverables are followed in organizations performing software maintenance. Evolution of the framework is also intended by incorporating the recommendations from the industry. Organizations that perform software maintenance are taken into consideration. Convenient sampling technique is used and the organization size varies from medium to large. Table 1 presents the units of analysis of the case study

Table 1: Units of analysis

Roles in SLA Management	Tenure of agreement	Scope of support	Specification of equipment required
The commitments and responsibilities of the parties involved in the SLA	Availability of support organizations	Definition of each task type in terms of monetary or non-monetary values	Rules for requesting and providing support
Prioritization of support tasks	SLA Templates	Process of negotiation	Design time repository
SLA Template registry	Process of service provisioning and deployment	Decomposing SLA into rules	Help desk

	SLA violation record and SLA adjustment patterns	Termination and decommission of the service	

Research questions addressed in the case study are:
RQ1: What are different aspects of a good SLA according to the industry?
RQ2: What are the attributes for a software maintenance specific SLA according to the industry?
RQ3: How do the practices in the industry vary from the prescribed practices in literature?
RQ4: What attributes and practices must be a part of maintenance SLA that can fill the gap between theory and practice?
Data collection techniques are divided into three levels:
• First degree: Direct contact with the subjects and collect data in real time interviews.
• Second degree: Indirect methods where we directly collect raw data without actually interacting with the subjects during the data collection.
• Third degree: Independent analysis of work artifacts where already available and compiled data will be used. Independent analysis of work artifacts is done where already available and compiled data is used.
In order to explore the preliminary framework in the industry through case study methodology (Runeson, Host, et al. 2012). The case study has been properly designed following the prescribed guidelines with an objective to gather as much information as possible from the industrial setting

6. Results from data collection

Case studies were conducted in five IT organizations, which perform software maintenance. The organizations that participated in the case studies are listed in Table II.

Table 2: Participating organizations*

Organization	Project Domain	Size	Process Model
A	Business Intelligence	Large	Agile
B	Health Records	Large	Agile
C	Insurance	Medium	Agile
D	Issuer Processing Solution	Large	Agile
E	Human Resource Management	Large	Agile

*Names of the organizations have been kept confidential

A. Quantitative Results

Quantitative techniques have been applied to the data collected through case studies. These have been elaborated as follows. we used practice performance rating measure [32]. Practice performance, PP(i), is rated by the maximum adequacy degrees among fully (F, 90 – 100%), largely (L, 60 – 89%), partially (P, 25 – 59%), and not (N, 10 – 24%) achieved, i.e.: PP(i) = max { F | L | P | N } = max { 5 | 3 | 1 | 0 } where i is the index number of a practice.
Assume a process, p, consists of mp practices. An average practice performance of the practices in a process, PP(p), can be derived by: PP(p) = 1 /mp = $\sum$ PP(i). Table III provides the practice performance rating for all the phases of SLA Management in studied IT Organizations
Performance of the following phases was evaluated through this method:

1. Developing SLA Template
2. Process of negotiations over SLA
3. Process of service deployment (Perform Transition to progressively transfer the system from the developer to the maintainer)
4. Maintain a help desk and decompose SLA into rules that can be monitored to further provide a way for analyzing the subsequent flow of monitoring events corresponding to these rules
5. Assessment and Corrective Actions over SLA
6. Decommissioning should be properly handled through SLA

The scores for the process of service deployment and assessment and corrective actions over SLA are low as shown in Table III. This shows a gap in industrial practice regarding these two practices.
We were interested in examining the applicability of our preliminary framework in the IT industry. For this purpose, we analyzed the compatibility of practices based on joint domain coverage [32]. According to compatibility definition, fork existing organizational frameworks, the compatibility degree, Ck, can be described at k levels. Where k=5, the five compatibility levels can be specified by; C1: activities that are only defined in specific organizational frameworks. Cn: activities identified in n of the organizational frameworks. Table IV provides key practice analysis for the practices identified in the preliminary framework.

Table 3: Practice performance rating

	A	B	C	D	E	PP(p)
Developing SLA Template	F	F	F	F	F	5
Process of negotiations over SLA is significant	F	F	F	F	F	5
Process of service deployment (Perform Transition to progressively transfer the system from the developer to the maintainer)	L	P	P	L	L	2.2
Maintain a help desk and decompose SLA into rules that can be monitored to further provide a way for analyzing the subsequent flow of monitoring events corresponding to these rules	F	F	F	F	F	5
Assessment and corrective Actions over SLA	P	P	P	P	P	1
Decommissioning should be properly handled through SLA	F	F	F	F	F	5

It is evident from Table IV that the concept of SLA registry and software landscape is not appreciated in the industry as prescribed in literature. Industrial practices regarding these were identified and the final framework has been modified accordingly. Literature was further explored for adjustment patterns and manageability for modifying SLA to give recommendations.

B. Qualitative Results

Subjective information was gathered while conducting the case study that applied to every phase of the framework. This provided enhancements to the framework and helped to shape the final framework. Service level management roles were identified, which are: Product owner, project manager, implementation lead, process manager, process analyst, client liaison and end user. Table IV elaborates SLA management roles.

SLA structure was defined that includes: Project name, personnel information, a tenure that includes not only dates but also rules of renewal and termination, client's desired outcome, communication channels: Focal persons, Modes of reporting, Complaint handling procedures, Surveys, Reviews.

Criticality identification where critical assets are identified and impact estimation caused by a loss of these assets is made, availability of stakeholders, required types and level of support: Onsite and offsite, commitments of continuity, specification of technical standards, responsibilities of stakeholders, pricing model, version control, annexures and references along with a glossary, if required. A stakeholder management process was identified which can be elaborated as a management process for stakeholders where they are initially identified, their interest and impact on the project are determined, communication channels established among them and their engagement is formally defined.

Scope definition and task prioritization process are also addressed. The scope is defined through objectives, goals, phases and sub-phases, responsibilities, funds, and financial plan while Priority and severity policy is applied where tasks are classified accordingly. It is observed that whether a task is critical, major, medium or low in terms of severity and where does it fall on the priority scale. In general, a severity-priority matrix is drawn and the task is placed in a particular quadrant.

Schedule SLA terms are initially negotiated by the product owner and the project manager. Once the process gets running implementation leads are involved in the negotiations.

In order to enable the maintenance team to perform its tasks, proper knowledge transfer has to be done. Following information sources are considered mandatory: Detailed specifications of the project, documentation (code and technical), credentials, procedures, and workflows.

SLA is decomposed through service level objectives (SLOs). SLOs need to be properly defined so that to ensure that they can be properly understood, attained, repeated, and controlled. They have to be affordable and acceptable for all the stakeholders. Incident management tools are used and a log of each and every incident is maintained based on a feature checklist. This log is further analyzed to perform any corrective action. Data migration and data archiving provisions are vital to the decommissioning phase and should be a part of the SLA.

Table 4: SLA Management Roles

Role	Description
Product owner	Product Owner is Responsible for defining, managing, controlling, and improving the SLA management process in order to ensure that the SLA management process and operational practices are both efficient and effective. He has to make sure stakeholders adequately participate in the SLA management process and keeps higher management well informed.
Project Manager	Product Owner is Responsible for defining, managing, controlling, and improving the SLA management process in order to ensure that the SLA management process and operational practices are both efficient and effective. He has to make sure stakeholders adequately participate in the SLA management process and keeps higher management well informed.
Implementation Lead	Heads the implementation of the processes described in the SLA and provides related information to the stakeholders. Reports discrepancies and facilitates process enhancement
Process Manager	Produces RFPs in coordination with stakeholders, makes vendor selections after evaluating proposals. He ensures that the process runs smoothly as he has to see if the vendors are properly providing the deliverables.
Process analyst	Responsible for production of reports, performing gap analysis, identifying SLA breaches, investigating causes of breaches and recommending remedial measures. It is his responsibility to coordinate review meetings and keep a track of actions taken accordingly.
Client Liaison	Represents the client
End User	The ones who actually consume the provided services

Table 5: Key Practice analysis

		A	B	C	D	E	
DST1	Identify all parties involved	1	1	1	1	1	C5
DST2	Mention Tenure of agreement	1	1	1	1	1	C5
DST3	Define scope of support	1	1	1	1	1	C5
DST4	Define Scope of the supported products is defined	1	1	1	1	1	C5
DST5	Specify equipment for providing support	1	1	1	1	1	C5
DST6	State the commitments and responsibilities of the parties involved in the SLA are stated	1	1	1	1	1	C5
DST7	State the availability of support organizations	1	1	1	1	1	C5
DST8	Define each support task type in terms of monetary and non-monetary terms	1	1	1	1	1	C5
DST9	State the rules for requesting and providing support	1	1	1	1	1	C5

DST10	Prioritize Support Tasks	1	1	1	1	1	C5
	SLA Templates						
NSLA1	Negotiate to agree upon the activities mentioned above in DST1 through DST10	1	1	1	1	1	C5
SDP1	Perform Transition to progressively transfer the system from the developer to the maintainer	1	0	1	0	1	C3
SPD2	Provision of SLA-specified services.	1	1	1	1	1	C5
	SLA Registry	0	0	0	0	0	C0
	The software Landscape	0	0	0	0	0	C0
ES1	Decompose SLA into rules that can be monitored and further provide a way for analyzing subsequent flow of monitoring events corresponding to these rules	1	1	1	1	1	C5
ES2	Establish a Help desk for Modification Requests and reporting problems:	1	1	1	1	1	C5
ES3	Accept or Reject modification requests	1	1	1	1	1	C5
ACA1	Record all the violations of SLA and initiate corrective Actions	1	1	1	1	1	C5
ACA2	Utilize adjustment patterns and model for manageability for modifying SLA	1	1	0	0	1	C3
DCM1	Decide the terms of decommissioning	1	1	1	1	1	C5

7. Refined framework

This research work started with the literature review and then grounded theory was applied to build a preliminary framework for SLA management for software maintenance. This framework was explored and evaluated in industry and enhancements were made to it. Summary of these enhancements is provided in Table VI. Refined phases of SLA management are presented.

A. Phase 1: Development of SLA Structure (DST)

All the necessary information regarding SLA is gathered in this phase. It provides the basis for any further process. Therefore, it is critical to developing a comprehensive SLA that misses none of the vital aspects of SLA management.

Table 6: Enhancements in the Preliminary Framework

Phase	Enhancements
Phase 1: Development of SLA Structure (DST)	1. Seven new activities identified 2. Three new deliverables identified
Phase 2: Negotiation of SLA (NSLA)	1. Concept of version control identified
Phase 3: Service provisioning and deployment (SPD)	1. Four more deliverables identified
Phase 4: Execution of the service (ES)	1. Two more deliverables identified
Phase 5: Assessment and corrective Actions (ACA)	1. One more deliverable identified
Phase 6: Decommissioning of service (DS)	1. Two more activities identified 1. Two more deliverables identified

The first phase of SLA is concerned with the initial drafting of SLA. It is important that all the stakeholders involved in software maintenance should mutually participate in developing an SLA (DST1). The stakeholders include the SLA Manager, SLA Administrator, and SLA Executor. The SLA Manager is the owner of SLA management process. SLA Administrator has the responsibility of facilitating and reviewing the SLA documentation. Finally, SLA Executor implements the services as decided in the SLA specifications

Table 7: Development of Service and Service Templates

Code	ACTIVITIES	Guidelines	Deliverables
DST1	Identify all parties involved	Following recommendations should be considered while mentioning service specifications: Efforts that would be made and their expected results must be specified Service specifications should be clear Service specifications should be complete There should be proper cost management SLA should not be a dead end document	SLA Structure Stakeholder management plan* Special agreements if applicable(eg; NDA)* Access Control List*
DST2	Mention Tenure of agreement		
DST3	Define the scope of support		
DST4	Define the Scope of the supported products is defined		
DST5	Specify equipment for providing support		
DST6	State the commitments and responsibilities of the parties involved in the SLA are stated		
DST7	State the availability of support organizations		
DST8	Define each support task type in terms of monetary and/or non-monetary values		
DST9	State the rules for requesting and providing support		
DST10	Prioritize support tasks		
DST11*	Communication between customer and service provider		
DST 12*	Service and asset criticality		

DST13*	Required types and levels of support		
DST14*	State Service level requirements/ targets		
DST15*	Mention Technical standards/ specification of the service interface		
DST16*	Provide a pricing model		
DST17*	Provide List of annexes and references and Glossary		

The stakeholders should decide on the time period for the provision of services (DST2). The third activity (DST3) is concerned with defining the scope of maintenance services. The projects that should be maintained are decided in advance (DST4). The maintenance team needs maintenance environment in order to provide effective services. Therefore, the requisition for maintenance equipment should be part of SLA (DST5). Every role participating in maintenance provision should have a clear understanding of his role and responsibilities so that future confusions can be avoided (DST6). It is important that the contact hours of all stakeholders should be mentioned in the SLA (DST7). It is important to categorize the support tasks on the basis of their monetary/time value (DST8). A communication mechanism should be formed for requesting and providing maintenance services (DST9). Maintenance tasks should be prioritized based upon their criticality (DST10). Enhanced activities include establishing communication channels among stakeholders (DST11), Criticality of services and assets is mentioned (DST12). Required types and levels of support are mentioned (DST13). Service level requirements and targets are mentioned (DST14). Technical standards and interface specifications are mentioned (DST15). A pricing model is provided (DST16). Annexure and glossary are provided where required (DST17). Enhanced deliverables include stakeholder management plan, special agreements if applicable, and access control lists. See Table VII.

B. Phase 2: Negotiation of SLA (NSLA)

This phase put emphasis on negotiation between stakeholders in order to form a consensus on the activities described in Phase 1 (DST). Two work products are produced as a result of this phase including (1) Design Time Repository and SLA Template Registry, that keeps a record of all the templates either possible or offered. A version control aspect is added to the SLA structure. The enhanced phase 2 of the framework is provided in Table VIII.

Table 8: Negotiation of SLA

Code	ACTIVITIES	Deliverables
NSLA1	Negotiate to agree upon the activities mentioned above in DST1 through DST17*	Design Time Repository SLA Template Registry Version Control*

C. Phase 3: Service provisioning and deployment (SPD)

Once an SLA is agreed upon, deployment is initiated and services provision begins. This phase comprises of activities and deliverables given in Table VIII This phase comprises of two activities. The first activity involves transitioning of the system from developer to maintainer (SPD1). The second activity is concerned with the provision of maintenance services by the maintainer (SPD2). Two work products are produced as a result of this phase. These are (1) SLA registry, that keeps a track of all agreed upon SLAs and (2) The Software Landscape that keeps a record of all products related to software for service provisioning in order to ensure that time-based and other types of service dependencies are being catered. Software landscape was elaborated in the industry by adding deliverables namely: project specifications, code documentation, development credentials, and deployment procedures. See table IX.

Table 9: Service provisioning and deployment

Code	Activities	Deliverables
SDP1	Perform Transition to progressively transfer the system from the developer to the maintainer	Project specifications* Code documentation* Development credentials* Deployment procedures*
SDP2	Provision of SLA-specified services.	

D. Phase 4: Execution of Service

Services mentioned in the SLA and deployed in the previous phase now become consistent. The initial set of rules may be evolved based on feedback from stakeholders (Activity ES1). The service provider establishes a help desk for accepting the modification requests submitted by service consumers (Activity ES2). Requests can be rejected or rerouted based on their size, required effort or the complexity (Activity ES3). The added deliverables are Service Level Objectives and Incident Management Log as mentioned in Table X.

Table 10: Execution of service

Code	ACTIVITIES	Deliverables
ES1	Decompose SLA into rules that can be monitored and further provide a way for analyzing the subsequent flow of monitoring events corresponding to these rules	Service Level Objectives*
ES2	Establish a Help desk for Modification Requests and reporting problems:	Incident Management Log*
ES3	Accept or Reject modification requests:	

E. Phase 5: Assessment and Corrective Actions

SLA management is an ongoing process and an SLA should be evaluated and updated iteratively. Once the execution has begun the assessment and correction process runs in parallel. It is significant to monitor service provision and record any SLA violations. The corrective actions must be initiated to rectify the shortcomings of SLA (ACA1). The added deliverables are Incident Management Analysis as mentioned in Table XI. Through Incident management analysis corrective actions are taken and SLA is modified or adjusted.

Table 11: Assesment and corrective actions

Code	ACTIVITIES	Deliverables
ACA1	Record all the violations of SLA and initiate corrective actions	Incident management analysis*
ACA2	Utilize adjustment patterns and model for manageability for modifying SLA	

F. Phase 6: Decommissioning of service

Decommissioning is the final phase of the SLA management life cycle where service is no longer provided. There should be a consensus on how the services will terminate and what would be the deliverables at the time of decommissioning. Decommissioning phase in the preliminary framework didn't provide much detail about the decommissioning process. However, in the refined framework data migration (DCM2) and data archiving (DCM3) are vital activities supported by their respective deliverables as mentioned in table XII.

Table 12: Decommissioning of Service

Code	ACTIVITIES	Deliverables
DCM1	Decide the terms of decommissioning	
DCM2*	Perform data migration	Data Migration Report*
DCM3*	Perform data archiving	Data Archiving Report*

Figure 4 graphically represents the refined software maintenance SLA management framework.

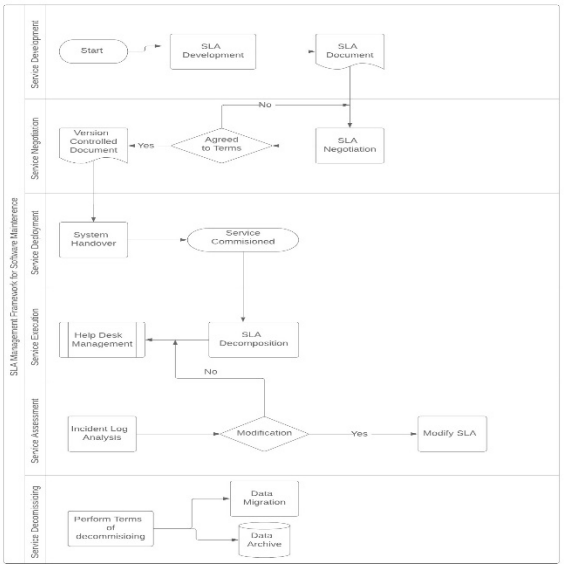

Fig. 5 Refined software maintenance SLA management framework

8. Conclusion

This paper presents an SLA Management Framework for software maintenance. We developed the framework in two major phases. In the first phase, we explored the state of the art in this domain and developed a preliminary framework. In the second phase, we evaluated and enhanced the framework by conducting industrial case studies. Every phase of the framework has been enhanced. SLA structure is refined and new concepts like service level objective and incident management analysis have been introduced, Decommissioning phase has been handled in detail. The final framework reflects industrial practice. We concluded that the majority of aspects mentioned in the framework are applicable in an industrial setting. However, gaps were identified in the process of service deployment, assessment and corrective actions over SLA and SLA modification.

Quantitative results showed that the scores for the process of service deployment and assessment and corrective actions over SLA are low. This shows a gap in industrial practice regarding these two practices. A process needs to be adapted to modify the SLA while the service is in progress rather than making ad-hoc measures and waiting for the tenure to end. The key practice analysis showed that the concept of SLA registry and software landscape is not appreciated in the industry as prescribed in literature. Industrial practices regarding these were identified and the final framework has been modified accordingly.

We found that the Development of Service Template is a major phase in maintenance SLA management. We found that stakeholder management, scope identification, and task prioritization are the key challenges of this phase.

Stakeholders must be identified and consulted before making any decision related to the SLA agreement. Regarding scope identification, we found that it is the main activity that decides about the effort and time required by the maintenance team to provide post-delivery maintenance services. Task prioritization depends on the criticality of each change requests; we found that it is important to group tasks under different categories based upon their criticality level. In order to meet the above-mentioned challenges, an SLA structure was further provided where vital clauses were identified for instance: Personnel information, tenure, client's desired outcome, communication channels, criticality identification, availability, required types and levels of support, targets, specification of technical standards, stakeholder responsibilities, pricing model, and version control were mentioned. Processes for stakeholder management, scope definition, and task prioritization were identified.

Regarding SLA Negotiation, we found that either the negotiation should be given enough time before moving on to the advanced phases or there must be a provision of modifying SLA during the service. In both cases, client and vendor have to be mutually comfortable.

As far as Service Provisioning and Deployment is concerned, it is important that a process of transition from developer to maintainer must be maintained that should include the elaborated and mandatory information sources: detailed specifications, documentation, credentials and procedure, and workflows.

Regarding Execution of service, it was identified that though assessment and corrective actions are taken in the process SLAs don't get radically modified during the process of service provision. It is also significant that the SLA is decomposed into achievable service level objectives.

There was an emphasis on the Decommissioning phase where generally there is a convention of elaborating decommissioning details. Organizations mostly archive data but there are organizations that further facilitate the client by assisting them in data migration as well.

References

[1] April, A., & Abran, A. (2009). A software maintenance maturity model (S3M): Measurement practices at maturity levels 3 and 4. Electronic Notes in Theoretical Computer Science, 233, 73-87.April, A., et al. (2005). "Software Maintenance Maturity Model (SMmm): the software maintenance process model." 17(3): 197-223.

[2] Boehm, B. W. (1981). Software engineering economics (Vol. 197). Englewood Cliffs (NJ): Prentice-hall.

[3] Bourque, P., & Fairley, R. E. (2014). Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.

[4] Cavalcanti, Y. C., da Mota Silveira Neto, P. A., Machado, I. D. C., Vale, T. F., de Almeida, E. S., & Meira, S. R. D. L. (2014). Challenges and opportunities for software change request repositories: a systematic mapping study. Journal of Software: Evolution and Process, 26(7), 620-653.

[5] Chatzimparmpas, A., & Bibi, S. (2019). Maintenance process modeling and dynamic estimations based on Bayesian networks and association rules. Journal of Software: Evolution and Process, e2163.

[6] Christa, S., Madhusudhan, V., Suma, V., & Rao, J. J. (2017). Software maintenance: from the perspective of effort and cost requirement. In Proceedings of the International Conference on Data Engineering and Communication Technology (pp. 759-768). Springer, Singapore.

[7] Comuzzi, M., Kotsokalis, C., Rathfelder, C., Theilmann, W., Winkler, U., & Zacco, G. (2009, November). A framework for multi-level sla management. In Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops (pp. 187-196). Springer, Berlin, Heidelberg.

[8] Cook, S. (2017). Measuring customer service effectiveness. Routledge..

[9] Hassan, S. I., & Khan, A. S. (2017, December). Eliciting Theory for Software Maintenance SLA Management Framework. In 2017 International Conference on Frontiers of Information Technology (FIT) (pp. 241-246). IEEE

[10] Hasselmeyer, P., Mersch, H., Koller, B., Quyen, H. N., Schubert, L., & Wieder, P. (2007, October). Implementing an SLA negotiation framework. In Proceedings of the eChallenges Conference (e-2007) (Vol. 4, pp. 154-161)

[11] Huang, H., Hu, M., Kauffman, R., & Xu, H. (2019, January). Renegotiation of Software Outsourcing Contracts. In Proceedings of the 52nd Hawaii International Conference on System Sciences..

[12] Jansson, A. S. (2007). Software maintenance and process improvement by CMMI. UPTEC STS07037 November Examensarbete, 20.

[13] Kajko-Mattsson, M. (2009, November). SLA management process model. In Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (pp. 240-249). ACM.

[14] Tsunoda, M., Monden, A., Matsumoto, K., Ohiwa, S., & Oshino, T. (2015, July). Benchmarking software maintenance based on working time. In 2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence (pp. 20-27). IEEE.

[15] Kajko-Mattsson, M., Ahnlund, C., & Lundberg, E. (2004, September). CM/sup 3: service level agreement. In 20th IEEE International Conference on Software Maintenance, 2004. Proceedings. (pp. 432-436). IEEE.

[16] Khan, A. S. (2013). A framework for software system handover (Doctoral dissertation, KTH Royal Institute of Technology).

[17] Lee, Y. J., & Choi, Y. J. (2015). A Study on the Information System Maintenance Activities and Performance. Journal of the Korea Society of Computer and Information, 20(12), 175-180.

[18] Lewis, L., & Ray, P. (1999). Service level management definition, architecture, and research challenges. In Seamless Interconnection for Universal Services. Global Telecommunications Conference. GLOBECOM'99.(Cat. No. 99CH37042) (Vol. 3, pp. 1974-1978). IEEE.

[19] Li, S. H., Yen, D. C., Lu, W. H., & Chen, T. Y. (2014). The characteristics of information system maintenance: an

empirical analysis. Total Quality Management & Business Excellence, 25(3-4), 280-295.
[20] McCalden, S., Tumilty, M., & Bustard, D. (2016, May). Smoothing the Transition from Agile Software Development to Agile Software Maintenance. In International Conference on Agile Software Development (pp. 209-216). Springer,
[21] Mikkonen, T., Lassenius, C., Männistö, T., Oivo, M., & Järvinen, J. (2018). Continuous and collaborative technology transfer: Software engineering research with real-time industry impact. Information and Software Technology, 95, 34-45.
[22] Monteiro, S. B. S., Lima, A. C. F., Venturini, F. C., & de Oliveira, W. S. (2018, September). Continuous improvement of systems in maintenance using proactive quality management. In 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC) (pp. 47-55). IEEE.
[23] Pigoski, T. M. (1996). Practical software maintenance: best practices for managing your software investment. Wiley Publishing.
[24] Reifer, D. J. (2016). Software Maintenance Success Recipes. Auerbach Publications..
[25] Rodríguez, P., Haghighatkhah, A., Lwakatare, L. E., Teppola, S., Suomalainen, T., Eskeli, J., ... & Oivo, M. (2017). Continuous deployment of software-intensive products and services: A systematic mapping study. Journal of Systems and Software, 123, 263-291.
[26] . Rodríguez, P., Haghighatkhah, A., Lwakatare, L. E., Teppola, S., Suomalainen, T., Eskeli, J., ... & Oivo, M. (2017). Continuous deployment of software-intensive products and services: A systematic mapping study. Journal of Systems and Software, 123, 263-291.
[27] Trienekens, J. J., Bouman, J. J., & Van Der Zwan, M. (2004). Specification of service level agreements: Problems, principles, and practices. Software Quality Journal, 12(1), 43-57.
[28] Trienekens, J. J., Bouman, J. J., & Van Der Zwan, M. (2004). Specification of service level agreements: Problems, principles, and practices. Software Quality Journal, 12(1), 43-57.
[29] Chapin, N., Hale, J. E., Khan, K. M., Ramil, J. F., & Tan, W. G. (2001). Types of software evolution and software maintenance. Journal of software maintenance and evolution: Research and Practice, 13(1), 3-30.
[30] Zhang, S., & Song, M. (2010, February). Architecture design of life cycle based SLA management. In 2010 The 12th International Conference on Advanced Communication Technology (ICACT) (Vol. 2, pp. 1351-1355). IEEE.
[31] Kung, H. J., & Hsu, C. (1998, November). Software maintenance life cycle model. In Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272) (pp. 113-121). IEEE.
[32] Wang, Y., & King, G. (2000). Software engineering processes: principles and applications. CRC press.