Application Lifecycle Management Activities For Quality Assurance In Software Development

Article · November 2017

CITATIONS
0

READS
411

4 authors, including:

Tobias Otibine
Kibabii University
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Samuel Mbuguah
Kibabii University ,Kenya, Bungoma
7 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Humphrey Juma Kilwake
Kibabii University College
3 PUBLICATIONS **0** CITATIONS
SEE PROFILE

International Journal of Trend in Research and Development, Volume 4(2), ISSN: 2394-9333
www.ijtrd.com

Application Lifecycle Management Activities For Quality Assurance In Software Development

[1]Tobias Okumu Otibine, [2]Samuel Mbuguah, [3]Juma Kilwake and [4]Harriet Loice Tsinale
[1,4]Department of Information Technology, [2,3]Department of Computer Science
[1,2,3,4]Kibabii University, Bungoma, Kenya

Abstract-- Lifecycle Management approaches promise more systematic and efficient ways to support the development and management of complex products. The concept of Application Lifecycle Management (ALM) indicates the coordination of activities and the management of artifacts during the software product's lifecycle. Most of the currently available unified ALM solutions are either based on basic version control and other 'low level' point-to-point integrations, or advocate the adoption of a new and expensive all-in-one solution from a single vendor. The problem with these current solutions is that the first does not go far enough to really provide the previously described benefits of applying an ALM solution, while the second one is often associated with high costs in tools, infrastructure and personnel which in turn affect the quality of software developed for SME's and middle level organizations. This paper discusses the effect of application lifecycle management activities on quality assurance in software development. The research employed multiple case study design. The data collection tools included Questionnaire, Observation and Interviews. ALM activities were found to be a direct predictor of Software quality assurance in software development. The research found out that ALM elements and their relations together with good documentation were very key in coming up with an efficient ALM solution and with improved process support and better knowledge and experience on ALM, application lifecycle management activities greatly affected the quality of software's developed thus improving on quality assurance in software development.

Keywords-- *Application Lifecycle Management; Software Development Lifecycle; Software Quality Assurance.*

I. INTRODUCTION

A. Background Information

In recent years, software has become more and more complex. In order to cope with this increased complexity, multiple levels of formal and logical abstractions had been introduced, each producing different set of artifacts representing a different view on the final product. All of the stages in the life of every software product are each supported by different development or product information management systems like requirements management databases, development tools, test tools, modeling tools, variability management tools, issue management tools, configuration management tools and others. Each of these tools targets only its very specific development lifecycle phase, however in order to successfully coordinate and control the software development activities and improve on quality of software developed, the artifacts and objects produced by these tools need to be interlinked and traced among each other for the purpose of process automation, reporting, impact analysis, regression testing, deployment and to help administer a cohesive and comprehensive software development process [1]. Unfortunately it is often very difficult to amalgamate all those different tools and to make them to exchange information, while still preserving the semantic and the consistency of the data, which introduces the challenge of finding a way to integrate all these different aspects of software development toward delivering high quality, long lasting and business critical products into the holy grail of modern software development.

This is where Application Lifecycle Management (ALM) comes into play. ALM provides an ecosystem of integrated tools, processes and domain technologies aimed at increasing the consistency, predictability and measurability of the software development process and beyond by coordinating, managing and keeping in sync the different involved activities as well as providing unification and automation for each of the major participating roles and stakeholders [2] by integrating development, collaboration, communication and knowledge management tasks and centralizing management of users, projects and processes. ALM can be the answer for the challenges of distributed software development, as it represents a giant leap towards controlled, manageable and auditable processes across all the different software development stages and beyond, which itself was an important step towards a full process automation and efficiency.

Although the main idea behind ALM, namely that a common infrastructure should exist, which helps to centralize, organize and align all software-related processes and tools and align their capabilities, is not new, it has only established itself as an autonomous discipline recently. In the past this idea of harmonizing the many software processes and tools had been pursued through tool integrations, which can be seen as the origin of ALM.

In the past decade, however, most of the ALM solutions available have been involved with the software development phases rather than quality and a majority of well off organizations using a full unified ALM solution which is not affordable for small and medium sized organizations. Thus the function of this paper is to find out how a small and medium sized organization can adopt the basic ALM activities that could aid in achieving quality assurance.

B. Statement of the Problem

Most of the currently available unified ALM solutions are either based on basic version control and other 'low level' point-to-point integrations, or advocate the adoption of a new and expensive all-in-one solution from a single vendor according to Georgi [3]. These ALM solutions do not consider the aspect of quality within them but just the phases of software development and how they could be integrated. The aspect of quality has been left vague. The purpose of this paper is to investigate on the various activities that if considered could greatly improve on quality of software developed by small and medium sized organizations.

II. RELATED STUDIES ON ALM ACTIVITIES FOR QUALITY ASSURANCE

A. Introduction

This section discusses related studies to the concept of ALM and ALM activities with their relation to quality assurance.

B. Application Lifecycle Management

In the past few years, the concept of Application Lifecycle Management has emerged to indicate the coordination of activities and the management of artifacts during the lifecycle of software products. ALM as a concept is quite new [4] and has mostly been discussed in professional articles, example, Doyle, [5], Schwaber, [6], Shaw, [7] and Carrillo, [2]. In the articles, the ALM concept has been discussed from various viewpoints, for instance:

1. Model-driven development [2]
2. Complex systems development [5]
3. Technology and ALM tools [8];[9];[10]; [11], or
4. Only treated in a cursory way [12]; [13].

According to Schwaber [6], companies are aware of ALM as a concept but do not understand what it actually means. The ALM tool providers have their own definitions of ALM that reflect their backgrounds and marketing strategies [14]. It is nonetheless difficult to find articles that discuss the concept: what constitutes ALM? This may affect the interpretation that the whole concept of ALM is unclear and driven by tool vendors Weiß [4], and Goth [8]. Pirklbauer et al, [14] argue that a wide range of tools are labeled as ALM tools due to the loosely defined scope of ALM. Similarly, Project lifecycle management (PLM) approaches have mainly been driven by tool providers and large user companies [15].

Most of the discussions relating to ALM as a concept are from professional publications: books and professional articles. There is a well-known professional article related to ALM from Schwaber [6]. She defines the three pillars of ALM to be traceability, process automation and reporting. An important view-point on ALM is that it does not focus on any specific lifecycle activity, but keeps all the activities synchronized [6]. ALM is therefore a thread that ties the development lifecycle together from business needs to operations [16].The support for project management has also been recognized. According to Doyle [5], a proper ALM tool provides strong support for project management by, for instance, providing an objective means to monitor project activities and generate real-time reports from project data. Schwaber [6] states that ALM does not necessarily require tools. Traditionally, lifecycle activities have been handled partly by manual operations.

Even though there is no general agreement on the definition of ALM, many concepts that have been defined as important to ALM have already been studied for a long time. Doyle [5], for instance, state that requirements management is important for ALM, and Schwaber [6] introduces the idea that concepts such as traceability, process automation, reporting and tool integration relate to ALM. Furthermore, the basis of ALM comes from Software Configuration Management (SCM). According to Weiß [4], the ALM tools have their roots in Configuration Management and Integrated Development Environments (IDEs). Murta et al., [17] and Schwaber [6] present SCM tools as the usual foundations of ALM infrastructures.

This therefore leads to our conclusion that ALM is the capability to integrate, coordinate and manage the different phases of the software delivery process. From development to deployment, it also involves a set of pre-defined processes and tools that include definition, design, development, testing, deployment and management.

Next, the issues that relate to ALM, such as requirements management, traceability, configuration management, tool integration and software quality assurance are discussed based on the literature.

a. Requirements Management and Traceability

Requirements management and traceability were under active research in the 90s by, for example, [18], [19], [20], [21], [22]). Sommerville and Sawyer [19] present guidelines for different requirements engineering process phases. In their book, requirements management is a support process for requirements engineering. They define the principal concerns of requirements management as managing changes to agreed requirements, managing relationships between requirements and managing dependencies between the requirements document and other documents. Kotonya and Sommerville [20] stress that requirements identification and storage are an essential pre-requisite of requirements management.

Requirements traceability (RT) refers to the ability to describe and follow the life of a requirement in a forward and backward direction [18]. A comprehensive study on reference models for requirements traceability is presented by Ramesh [21]. More generally, traceability deals with the relations between any lifecycle artifacts. Gills [23] presents a summary of the survey on traceability models in industrial projects. The survey summarizes industrial experiences, item types and traceability models used in industry. Gills [23] also presents a traceability model, based on the survey data, with the most typical items and relations. His study shows that each organization has its own needs and terminology for traceability. This leads to the need for traceability tailoring [24]. Espinoza and Garbajosa [25] have approached the problem of project-specific traceability requirements with traceability metamodels that include a basic set of items (concepts, structures) for project-specific extensions.

b. Configuration Management

Configuration Management and, more specifically from a SW management point of view, Software Configuration Management (SCM) is a discipline that provides the processes and technologies to identify and control (configuration) items [26]. SCM as a discipline has been in existence for several decades [27]. It is an important concept for ALM, as SCM tools can be seen as the foundations of ALM infrastructures [6].

The generic CM process contains the basic CM activities (configuration identification, configuration control, configuration status accounting and CM planning [7]. Swebok, [28] extends the basic CM activities with release management and delivery activity, which refers to the distribution of a software configuration item outside the development activity. Among the CM activities, the configuration identification activity provides a basis for other CM activities [28]. Shaw [7] has presented configuration management activities as a chronological process. In the process, previous steps form the basis of successive steps. Identification activities, for instance, are used to establish and maintain a definitive basis for control (i.e., Change Management) and status accounting. Configuration management planning provides mechanisms for planning and documenting the CM solution for a project. IEEE Std-828 assists in the planning of software CM by providing pre-structured templates for documenting the responsibilities and practices. Estublier, [27] divides the basic functionality of the SCM tools into three main classes: repository for components, help for engineers' usual activities and process control and support. Estublier et al. [27] present SCM as one of the software engineering domains in which process support has

proven to be most successful. Schwaber [6] present that customizable process templates in process-centric SCM tools provide ability to implement different processes for different projects. Software configuration management tools are typically file based, meaning that the granularity of the management is the file (version). For instance, the requirements document may contain several requirements as paragraphs. SCM tool can treat this document as a single aggregate object that contains all the requirements [29]. If a requirements specification is prepared as one document, it is possible to manage different versions of it, but it is not possible to control the requirement items separately [30]. This means that the assignment and maintenance of traceability information are more difficult compared with the fine-grained management of requirements in which each requirement is treated as its own object. Nowadays, requirements management tools manage atomic requirements in a requirements database and, therefore, allow the management of relationships between them. The challenge, however, lies in mechanisms to handle the traceability of lifecycle artifacts that reside in other databases, such as, test cases, test data, design elements, and source code. This has been a deficiency of configuration management tools [31], and ALM tools need to fix this.

C. Software Quality Assurance

This is the function of software quality that assures that the standards, processes, and procedures are appropriate for the project and are correctly implemented. There are many definitions of these Software Quality Attributes but a common one is the FURPS+ model which was developed at Hewlett Packard. Under the FURPS + model, the following characteristics are identified:-

a. Functionality

The F in the FURPS+ acronym represents all the system-wide functional requirements that we would expect to see described. These functional requirements represent the main product features and answer the question: What does the product do for us rather than how does it do it. The easiest way to think of functional requirements is to ask the question; why does this piece of software exist. This question of reason for being is distinct from security, look and feel and reliability concerns which are important but are not concerned with the main function of the software.

b. Usability

Usability includes looking at, capturing, and stating requirements based around user interface issues, e.g. issues such as accessibility, interface aesthetics, and consistency within the user interface.

c. Reliability

Reliability includes aspects such as availability, accuracy, and recoverability, for example recoverability of the system from shut-down failure.

d. Performance

Performance involves issues such as throughput of information, system response time (which also relates to usability), recovery time, and startup time.

e. Supportability

This is a general bucket of requirements that address supporting the software: for example testability, adaptability, maintainability, compatibility, configurability, installability, scalability, and localizability.

The "+" of the FURPS+ acronym allows for the specification of constraints, including design, implementation, interface, and physical constraints. The specification of the FURPS+ characteristics needs to go into the Systems Requirements. The testing of these characteristics is usually done by the SQA (testing team). Some of the FURPS+ characteristics, i.e. Functionality and Usability can be tested by executing the actual software.

Some, however, like Supportability and Adaptability can only be verified by code inspection or dry running What if' scenarios.

In this research, the FURPS + model will be used to test on Software Quality Assurance of Developed software products.

III. METHODOLOGY

A. Research Design

The research employed a multiple case study design since the research dealt with a detailed intensive study of the ALM cycle and tool integration within different firms. Case Study is of particular interest when the purpose is to gain a rich understanding of the context in the research and the processes being enacted. Yin [32] affirms that compared to other methods, the strength of the case study method is its ability to examine, in-depth, a case within its real life context. Polit [14] characterize case studies by their focus on one or few instances, their ability to highlight important areas of research due to their in-depth study nature, their power in explaining relationships, causes, and processes rather than relying only on outcomes. The research used qualitative and quantitative approach to carry out the study because this was a problem centered study.

The research was carried out in three phases within software development industries. In the first phase, a thorough literature review of scientific articles and books to understand the domain backgrounds and to seek out the theoretical setbacks that ALM poses was done.

The second step concentrated on carrying out a stakeholder analysis in order to identify the major stakeholders of an ALM solution. A scientific study was conducted to identify their requirements and to better understand their problems with ALM.

Thirdly, in order to identify what the top ALM products offer and lack, a scientific evaluation was performed on the most popular ALM tools against criteria derived directly from the elicited stakeholder requests and requirements.

a. Location of study

The study took its sample population from software development industries in Eldoret Town, Uasin-gishu County. The target population was the various users involved with ALM activities in software development from the software development firms within the county. The accessible organizations were Ronsmart limited, TopAfriq Ltd and Flexcom.

The reason to why the study was based on software firms in Eldoret Town, Uasin Gishu County, was because of the rise of the number of small and medium size software development firms coming up within Eldoret and since quality is a very important aspect within software development, this study would come up with better ways of improving on the same aspect.

b. Study Population

Polit [14], defines a population as the entire aggregation of cases that meet a designated set of criteria. The target population for this study was the various professionals involved with ALM activities in software development industries within Eldoret Town, Uasin-gishu County. The target population included Managers, Developers, Testers, Analysts, Directors and Architects.

c. Frame and Sample Size

Choosing study sample is an important step in any research project since it is rarely practical, efficient or ethical to study whole populations.
The sampling method the research employed for the study was purposive sampling. Purposive sampling refers to situations where participants are selected based on their specialized insight or special perspective, experience, characteristic, or condition that we wish to understand [33]. This enabled the research to pick only the population that had ALM activities integrated within software development stages. The target population constituted of 10 Managers, 60 Developers, 60 Testers, 30 Analysts, 20 Directors and 20 Architects in all the organizations since this was a case centered study.

Table 1

Population	Groups	Target population
Company Staff	Managers	10
	Developers	60
	Testers	60
	Analysts	30
	Architects	20
	Directors	20
Total		200

Sample Frame (Source: Author)

The sample size was calculated at 95% confidence level, an alpha level of 0.05 which was margin of error of +/- 5% and .5 as the standard of deviation which shows how much variance the research expected from the responses.
According to Cochran,

$$n_0 = \frac{z^2 pq}{e^2}$$

Where:

z – Z- score
p - Standard Deviation. Denoted as σ
q = 1 − σ
e - Margin of error

$$n_0 = \frac{(\text{Z} - \text{score})^2 * [\,\sigma * (1 - \sigma)]}{e^2}$$

Where:

n_o – representative sample
Z-score = 1.96 which was 95% confidence level
σ – Standard Deviation /Maximum variability - the research will expect a variance of .5 in its responses
e- Margin of Error (Confidence Interval) of ± 5%

$$n_0 = \frac{(1.96)^2 * [0.5 * (1 - 0.5)]}{(0.05)^2}$$

$$= \frac{3.8416 * [(0.5 * (1 - 0.5)]}{0.0025}$$

$$= \frac{3.8416 * [0.5 * 0.5]}{0.0025}$$

$$= \frac{0.9604}{0.0025}$$

$$= 384.16$$

$$= 385$$

According to Cochran [14], since the population is below 50,000 then the sample size is calculated by:

$$n = \frac{n_0}{1 + \frac{n_0 - 1}{N}}$$

Where:
N – Target population
n – Sample size

$$= \frac{385}{1 + \frac{385 - 1}{200}}$$

=131

The sample size for the study was calculated at 131 participants.
Managers were selected because they are the ones who come up with the final decisions on which methodologies to follow and which tools to use. The developers were chosen because they play a critical role of implementing design from system designers. The system testers were chosen because they function to verify that software's are developed basing on specifications thus ensuring quality. System analysts were chosen because they play a major role in conversion of user specification to a form in which designers of the system can understand. The system designers convert the documentation from analysts to design. The users were the people who used the developed system.

B. Data Collection Instruments

The research employed mixed techniques such as questionnaires, interview, observations and analysis of past literature on ALM solutions as data collection instruments. Sandelowski [34] states that the set of concrete operations at the technique level of research entail the combined use of data collection techniques that were commonly (but not necessarily) associated with either qualitative or quantitative research, such as open-ended and un-structured interviewing and structured questionnaires, respectively. Through these techniques it enabled the research be able to collect all the required data which was analyzed to give accurate result.

a. Questionnaire

Questionnaires were used because they gave a detailed answer to complex problems and they were also used for the purpose of meeting the objectives of the study. Questionnaires were administered simultaneously to large groups thus allowing the questions to reach a given number of respondents more efficiently and this will be saving on time and cost.

The questionnaires contained both structured (closed-ended) and unstructured (open-ended) questions. Open and closed-ended questions were used to elicit qualitative and quantitative information from the employees. The questionnaire items on the closed ended were measured using the 3-point and 5-point Likert scales.

The questionnaires provide a standardized data-gathering procedure. Using a well-constructed questionnaire can minimize the effects of potential human errors for example, altering the pattern of question that were asked, calling at inconvenient times, and biasing by "explaining.

Another advantage to why questionnaires were used was that the respondent will not have anyone to impress with his/her answers and need not have any fear of anyone hearing them.

Most surveyors believe the respondent will answer a questionnaire more frankly than he would answer an interviewer, because of a greater feeling of anonymity [16]. To maximize this feeling of privacy, it was important to guard, and emphasize, the respondent's anonymity.

b. Interviews

Interviews were used in this study since they generally yield highest cooperation and lowest negative response rates, offers high response quality and takes advantage of interviewer presence and it was multi-method data collection in that it combined questioning, cross-examination, probing techniques [35].

The interviews were conducted face-to-face with all the involved stakeholders. An interview schedule with a list of questions that the researcher used during the interview ensured good use of limited interview time. Both structured and non-structured interview were used. At first, unstructured interview was used so as to limit the risk of missing valuable information since the interviewee was able to talk freely about the subject and was not obstructed by any predefined agenda [36] then unstructured interview, followed which meant that the interview was based on a predefined set of questions. The interview guide was employed to help the researcher draw out verbal responses from the IT personnel's and the managers.

C. Quality Control

This research applied Reliability and Validity to ensure quality of results being found by the research tools.

a. Reliability of the Instruments

The internal consistency reliabilities of the summated scale variables were tested with Cronbach's Alpha coefficient and that they were not below 0.70, according to recommendations by Santos [37]. The items with insufficient loadings were not included to the summated scale variables so as to increase consistency. Cronbach's Alpha correlation was used to determine how items correlate among themselves [36].

This is shown in Table II.

Table 2

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Cronbach's Alpha if Item Deleted
ALM is important for your organization?	73.99	76.669	.486	.795
ALM in all its facets is feasible for s mall and middle-sized companies?	75.29	72.616	.412	.796
Does your organization currently use an ALM solution?	74.08	81.010	.093	.809
ALM solution is complete	77.24	77.148	.420	.797
Requirement management module is necessary in ALM	74.08	82.121	.012	.811
Modeling and system design activity necessary in ALM	74.09	74.288	.550	.790
Software configuration management activity is necessary in ALM	74.19	75.378	.425	.795
Test management activity is necessary in ALM	73.94	78.378	.408	.799
Defect management activity is necessary in ALM	73.94	79.971	.239	.804
Release management module is required in ALM	74.14	77.988	.404	.798
ALM is important for your organization	73.83	80.802	.261	.804
One should be able to trace what an application does to its requirements	74.29	80.718	.061	.813
The interface of the application developed should be easy to work with and maneuver around.	76.79	75.269	.246	.808
The developed application should always be available, accurate, and recoverable in case of any failure.	76.69	70.843	.631	.783
The applications throughput of information, system response time, recovery time, and startup time should always tested.	74.55	73.945	.347	.800
The application developed should be testable, can adapt to different environments, maintainable, compatible with different hardware and operating systems, configurable, installable, scalable, and localizable.	76.79	73.303	.428	.795
It is important to have an SQA team in application development	73.94	79.022	.339	.801

Reliability coefficients 17 items

Alpha = 0.806 Standardized item alpha = 0 .807

3.3.2 Validity of the Instruments

This section validates the correctness of the results that were achieved.

Validity refers to the degree to which an instrument measures what it intends to measure [35].

In order to test the validity of the instruments, first the questionnaires were scrutinized and approved by the university

supervisor and a group of experts before issuing them. Then a pretest of the instruments was done by piloting in two companies which were not part of the study. The pre-test was done so as to test whether the questions were clear or not to the respondents. It also tested the ambiguity of words that could bring about wrong interpretations of questions. After piloting, the ambiguous questions were corrected and the questionnaires issued back to the same respondents. This was done to determine whether the instrument would yield the required data.

D. Data Analysis

The data that was collected was first edited so as to remove errors then coded before being entered into the SPSS and Smart PLS for analysis. The Data was analyzed according to descriptive information from the questionnaire and following the research questions. Descriptive statistical analysis was employed to enable the research reduce, organize, summarize, evaluate, interpret and present the numeric information. Descriptive statistics makes it easy to analyze and it's convenient for the research and the study. It was in form of means, percentages and frequency distribution.
The findings were presented in tables, mean, percentages and frequencies Karl Pearson's coefficient of correlation was used as an inferential statistic to determine the relationship between the dependent and independent variable.

E. Ethical Issues

As this study required the participation of an organization involved with software development of critical applications, consent and confidentiality was addressed. The respondents were also advised that they could withdraw from the study even during the process. With this, the organization was not forced to participate in the research. The confidentiality of the organization was also ensured by not disclosing their information in the research. Only relevant details that helped in answering the research questions were included.

The researcher then got a research permit from the National Commission for Science, Technology and Innovation (NACOSTI) to carry out research.

IV. RESULTS PRESENTATION AND DISCUSION

A. Demographic Information of the respondents

This subsection presents and discusses the demographic information of respondents as categorized by interest in findings, age bracket, gender, age group, role and size of organization.

a. Respondents as categorized by Interest in findings

The study obtained information on the interest of the findings by the respondents, which has been presented in Table III.

Table 3

	Frequency	Percent	Valid Percent	Cumulative Percent
Valid Yes	120	100.0	100.0	100.0

Respondents by Interest in findings

It was established that all of the respondents were interested in the findings of the study. This therefore implies that all the respondents were interested in the results of the research.

b. Respondents as categorized by gender

The study obtained information on the gender of the respondents, which has been presented in Figure I.
A total of 120 participants completed questionnaires. This consisted of 35% (42) female and 65 % (78) male. It can be seen from Figure 4.1, that majority of the respondents were male. This therefore implies that findings obtained may not strongly capture the unique features of the female gender.

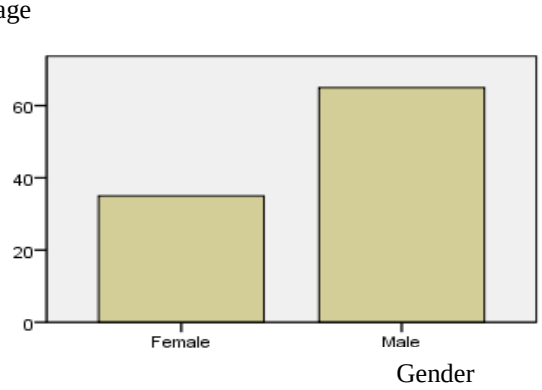

Figure 1: Respondents categorized by gender

c. Respondents categorized by age

Age in years
Table IV presents demographic information pertaining the age of the respondents.

Table 4

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	18 - 25 years	12	10	10	10
	26 - 33 years	60	50	50	60
	34 - 41 years	23	19.2	19.2	79.2
	42 - 50 years	14	11.7	11.7	90.8
	Over 50 year	11	9.2	9.2	100
	Total	120	100	100	

It was established that majority of the respondents were 26-33 years old. This formed 50% (60) of all respondents. It was also noted that other age groups formed a small percent of the remaining respondents. This were age groups 18–25, 34-41, 42-50, and over 50 years old formed a percentage of 10% (12), 19.2% (23), 11.7% (14) and 9.2% (11) respectively. This therefore implies that the findings obtained may strongly capture the views of the age group 26-33 years.

d. Respondents categorized by Designation

The study established Designation of the respondents by the roles they fulfill in organizations. The finding on this is tabulated in Table V. From the tabulated findings, it can be seen that majority of the respondents were developers and Testers who formed 25% (30) each with 5.0% (6) females and 20.0% (24) males and 15.0% (18) females and 10.0% (12) males respectively, followed by Analysts and Architects at 15% (18) with 0% (0) females and 15% (18) males and 10% (12) females and 5% (6) males respectively. And lastly Managers

and Directors at 10% each with 0.0% (0)females and 10.0% (12)males and 5.0% (6) females and 5.0% (6) males respectively. This therefore implies that the findings obtained may strongly capture the views of the all the specific roles with Developers and testers taking a bigger percentage because of their major role in building and ensuring quality on applications.

Table 5

		What is your gender?		
		Female	Male	Total
Development Manager / Other IT Manager	Count	0	12	12
	% of Total	0.0%	10.0%	10.0%
Designer / Developer	Count	6	24	30
	% of Total	5.0%	20.0%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	18	12	30
	% of Total	15.0%	10.0%	25.0%
Business Analyst / Requirements Engineer	Count	0	18	18
	% of Total	0.0%	15.0%	15.0%
CIO / IT Director / Board Director	Count	6	6	12
	% of Total	5.0%	5.0%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count	12	6	18
	% of Total	10.0%	5.0%	15.0%
Total	Count	42	78	120
	% of Total	35.0%	65.0%	100.0%

Respondents by Designation

e. Respondents categorized by Experience in their current designation

Table VI presents demographic information pertaining the experience of I.T experts in their current designation.
It was established that most of the specialists had 3 to 9 years of experience totaling to 71.7%, basing on the results where 3-5 years' experience consisted of 41.7% while 6-9 years had 30%. It was also established that less staff had experience of below 2 years (17.5%) and above 9 years (10.8%).This suggests that responses received would very well provide most accurate results since most of the I.T employees have enough experience in their various fields of expertise.

Table 6

		Indicate the number of Years in the current designation				
		Below 2 years	3-5 years	6-9 years	above 9 years	Total
Development Manager / Other IT Manager	Count	3	3	4	2	12
	% of Total	2.5%	2.5%	3.3%	1.7%	10.0%
Designer /Developer	Count	4	16	8	2	30
	% of Total	3.3%	13.3%	6.7%	1.7%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	7	12	8	3	30
	% of Total	5.8%	10.0%	6.7%	2.5%	25.0%
Business Analyst / Requirements Engineer	Count	2	7	7	2	18
	% of Total	1.7%	5.8%	5.8%	1.7%	15.0%
CIO / IT Director / Board Director	Count	2	4	4	2	12
	% of Total	1.7%	3.3%	3.3%	1.7%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count	3	8	5	2	18
	% of Total	2.5%	6.7%	4.2%	1.7%	15.0%
Total	Count	21	50	36	13	120
	% of Total	17.5%	41.7%	30.0%	10.8%	100.0%

Respondents categorized by Experience

f. Respondents categorized by size of the organization

Table VII presents demographic information pertaining the size of the organization.

Table 7

	Frequency	Percent	Valid Percent	Cumulative Percent
Less than 10 employees	24	20.0	20.0	20.0
10 to 49 employees	54	45.0	45.0	65.0
50 to 249 employees	42	35.0	35.0	100.0
Total	120	100.0	100.0	

Size of the Organization

It was established that most of the organizations had employees of between 10 to 49 leading to 45% followed by 50 to 249 (35%) and lastly less than 10 employees (20%). This therefore implies that the findings obtained may strongly capture the views of small and mid-sized organizations.

B. ALM Activities

The study sought to find out how the respondents perceived ALM activities as a determinant for Quality assurance in software development.
This section reports on results of the ALM elements and their importance from the various participants involved. This formed the independent variable in the study. To ascertain the inputs of ALM elements, the mean of each input was calculated on a 5 point scale where "1 = Not Important", "2 = Less Important", "3 = Not sure", "4 = Important" and "5 = Very Important".

a. Importance of ALM to an organization

Table VIII presents results on views of the importance of ALM to an organization from different experts.
Table VIII.

		ALM is important for your organization		
		Agree	Strongly Agree	Total
Development Manager / Other IT Manager	Count	0	12	12
	% of Total	0.0%	10.0%	10.0%
Designer / Developer	Count	0	30	30
	% of Total	0.0%	25.0%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	0	30	30
	% of Total	0.0%	25.0%	25.0%
Business Analyst / Requirements Engineer	Count	12	6	18
	% of Total	10.0%	5.0%	15.0%
CIO / IT Director / Board Director	Count	6	6	12
	% of Total	5.0%	5.0%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count	0	18	18
	% of Total	0.0%	15.0%	15.0%
Total	Count	18	102	120
	% of Total	15.0%	85.0%	100.0%

Respondents categorized by importance of ALM to an organization

This question sought to find out what the different experts thought of ALM as being important to their organizations. On responding to the question, 85.0% strongly agreed and 15% agreed that ALM was important as shown in Table VIII. This totaled to 100% meaning that all the experts were of the opinion that ALM is a necessity to software development management.

b. Importance of ALM to a specific designation

Table IX presents results on views of the importance of ALM to a specific designation from different experts.
Table IX.

		ALM is important for your particular role in the organization			
		Not Sure	Agree	Strongly Agree	Total
Development Manager / Other IT Manager	Count	0	6	6	12
	% of Total	0.0%	5.0%	5.0%	10.0%
Designer / Developer	Count	6	0	24	30
	% of Total	5.0%	0.0%	20.0%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	0	0	30	30
	% of Total	0.0%	0.0%	25.0%	25.0%
Business Analyst / Requirements Engineer	Count	6	6	6	18
	% of Total	5.0%	5.0%	5.0%	15.0%
CIO / IT Director / Board Director	Count	0	0	12	12
	% of Total	0.0%	0.0%	10.0%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count	0	0	18	18
	% of Total	0.0%	0.0%	15.0%	15.0%
Total	Count	12	12	96	120
	% of Total	10.0%	10.0%	80.0%	100.0%

Respondents categorized by importance of ALM to a specific designation

This question sought to find out what the different experts thought of ALM as being important to their specific roles in their organizations. On responding to the question, 80.0% strongly agreed 10% agreed and 10% were not sure as shown in Table IX. It was also seen that 5% of the Managers strongly agreed and 5% agreed bringing to a total of 10% which is the total population for Managers. 20% of developers strongly agreed and 5% were not sure. 25% of Testers strongly agreed, 5% of analysts strongly agreed while 5% agreed and 5% were not sure bringing to a total of 10% who agreed, 10% of directors totally agreed which is the total population for directors and lastly 15% of architects strongly agreed which is their total population. With the results above it suggests that ALM is very important for the different roles involved in the SDLC.

c. ALM in all its facets is feasible for small and middle-sized companies

Table X presents results on views of experts on if ALM in all its facets is feasible for s mall and middle-sized companies. This question sought to find out whether ALM was feasible for small and medium sized organizations. On responding to the question, 10.0% strongly agreed 55% agreed bringing to a total of 65% of those who agreed, 15% were not sure, 5% did not agree while 15% strongly disagreed bringing to a total of 20% who did not agree. Thus it can be seen that a majority of respondents agreed that ALM could work for small and mid-sized organizations. A good number disagreed with the response of believing that ALM could only work on large organizations attributed to its complex nature and cost.

Table 10

		ALM in all its facets is feasible for s mall and middle-sized companies					
		Strongly Disagree	Disagree	Not Sure	Agree	Strongly Agree	Total
Development Manager / Other IT Manager	Count	0	0	0	12	0	12
	% of Total	0.0%	0.0%	0.0%	10.0%	0.0%	10.0%

Designer / Developer	Count	0	0	2	28	0	30
	% of Total	0.0%	0.0%	1.7%	23.3%	0.0%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	2	0	10	8	10	30
	% of Total	1.7%	0.0%	8.3%	6.7%	8.3%	25.0%
Business Analyst / Requirements Engineer	Count	16	0	0	2	0	18
	% of Total	13.3%	0.0%	0.0%	1.7%	0.0%	15.0%
CIO / IT Director / Board Director	Count	0	2	6	4	0	12
	% of Total	0.0%	1.7%	5.0%	3.3%	0.0%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count	0	4	0	12	2	18
	% of Total	0.0%	3.3%	0.0%	10.0%	1.7%	15.0%
Total	Count	18	6	18	66	12	120
	% of Total	15.0%	5.0%	15.0%	55.0%	10.0%	100.0%

Respondents categorized by feasibility of ALM to small and mid-sized organizations

d. Use of ALM solution

Table XI presents results on views of experts on their organizations use of ALM solution.

Table 11

		Do you apply ALM in your organization			
		Not Sure	Agree	Strongly Agree	Total
Development Manager / Other IT Manager	Count	0	6	6	12
	% of Total	0.0%	5.0%	5.0%	10.0%
Designer / Developer	Count	0	0	30	30
	% of Total	0.0%	0.0%	25.0%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	6	0	24	30
	% of Total	5.0%	0.0%	20.0%	25.0%
Business Analyst / Requirements Engineer	Count	0	12	6	18
	% of Total	0.0%	10.0%	5.0%	15.0%
CIO / IT Director / Board Director	Count	6	6	0	12
	% of Total	5.0%	5.0%	0.0%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count	0	0	18	18
	% of Total	0.0%	0.0%	15.0%	15.0%
Total	Count	12	24	84	120
	% of Total	10.0%	20.0%	70.0%	100.0%

Respondents categorized by use of ALM solution

This question sought to find out whether the different organizations use ALM solutions. On responding to the question, 70.0% strongly agreed 20% agreed bringing to a total of 90% of those who agreed and 10% were not sure. Thus it can be seen that a majority of respondents agreed that they use an ALM solution in their organization and thus giving a good population for the study although the 10 percent were not aware of the ALM.

4.2.5 Complete ALM solution

Table XII presents results on views of experts on their organizations use of ALM solution.

Table XII.

		ALM solution is complete			
		Strongly Disagree	Disagree	Not Sure	Total
Development Manager / Other IT Manager	Count	6	6	0	12
	% of Total	5.0%	5.0%	0.0%	10.0%
Designer / Developer	Count	18	12	0	30
	% of Total	15.0%	10.0%	0.0%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	6	12	12	30
	% of Total	5.0%	10.0%	10.0%	25.0%

Business Analyst / Requirements Engineer	Count	18	0	0	18
	% of Total	15.0%	0.0%	0.0%	15.0%
CIO / IT Director / Board Director	Count	12	0	0	12
	% of Total	10.0%	0.0%	0.0%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count	18	0	0	18
	% of Total	15.0%	0.0%	0.0%	15.0%
Total	Count	78	30	12	120
	% of Total	65.0%	25.0%	10.0%	100.0%

Respondents categorized by complete ALM solution

This question sought to find out whether the ALM solution that they use is a complete solution covering all the ALM activities. On responding to the question, 65.0% strongly disagreed 25% disagreed bringing to a total of 90% of those who disagreed and 10% were not sure. This shows that most of the organizations have ALM solutions but are not complete in that they offer specific ALM services but not all thus they are not satisfied with their solutions. This shows that they would like more to be done on the ALM solution.

4.2.6 Requirement management module

Table XIII presents results basing on the recommendation of the different experts on the ALM requirement management module.

Table XIII.

		Requirement management module is necessary in ALM			
		Not Sure	Agree	Strongly Agree	Total
Development Manager / Other IT Manager	Count	0	12	0	12
	% of Total	0.0%	10.0%	0.0%	10.0%
Designer / Developer	Count	0	6	24	30
	% of Total	0.0%	5.0%	20.0%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	0	6	24	30
	% of Total	0.0%	5.0%	20.0%	25.0%
Business Analyst / Requirements Engineer	Count	0	12	6	18
	% of Total	0.0%	10.0%	5.0%	15.0%
CIO / IT Director / Board Director	Count	6	0	6	12
	% of Total	5.0%	0.0%	5.0%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count	0	0	18	18
	% of Total	0.0%	0.0%	15.0%	15.0%
Total	Count	6	36	78	120
	% of Total	5.0%	30.0%	65.0%	100.0%

Respondents categorized by requirements management module

From the results depicted in the table above (Table XIII), most of the experts agreed that the Requirement management module was very important in ALM. 65.0% of the respondents strongly agreed that it was very important while 30.0% agreed that it was important bringing to the total of 95.0% of those who agreed. This is because the requirements management module gives an organization a single shared repository to collaborate and share requirements, understand their relationship to tests, and evaluate linked defects among the requirements thus its importance in the SDLC of an application. It is also noted that all (15%) of the analysts advocated for the requirements management module since it forms part of their day to day activities thus a necessity for them.

4.2.7 Modeling and system design activity is necessary in ALM

Table XIV presents results basing on the recommendation of the different experts on modeling and system design activity.

Table XIV.

		Modeling and system design activity is necessary in ALM				Total
		Disagree	Not Sure	Agree	Strongly Agree	
Development Manager / Other IT Manager	Count	0	0	0	12	12
	% of Total	0.0%	0.0%	0.0%	10.0%	10.0%
Designer / Developer	Count	0	0	0	30	30
	% of Total	0.0%	0.0%	0.0%	25.0%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	0	6	0	24	30
	% of Total	0.0%	5.0%	0.0%	20.0%	25.0%
Business Analyst / Requirements Engineer	Count	6	0	12	0	18

Analyst / Requirements Engineer	% of Total	5.0%	0.0%	10.0%	0.0%	15.0 %
CIO / IT Director / Board Director	Count	0	0	0	12	12
	% of Total	0.0%	0.0%	0.0%	10.0%	10.0 %
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count	0	0	6	12	18
	% of Total	0.0%	0.0%	5.0%	10.0%	15.0 %
Total	Count	6	6	18	90	120
	% of Total	5.0%	5.0%	15.0%	75.0%	100.0 %

Respondents categorized by modeling and system design activity

This question sought to find out whether modeling and system design activity is necessary in ALM. On responding to the question, most of the experts agreed that the modeling and system design activity was very important in ALM. 75.0% of the respondents strongly agreed that it was very important while 15.0% agreed that it was important bringing to the total of 90.0% of those who agreed. 5% were not sure while 5% disagreed. With these results it shows that modeling and system design activity is very key in ALM and thus must always be included.

4.2.8 Software configuration management activity is necessary in ALM

Table XV presents results basing on the recommendation of the different experts on software configuration management activity.

Table XV.

		Software configuration management activity is necessary in ALM				
		Disagree	Not Sure	Agree	Strongly Agree	Total
Development Manager / Other IT Manager	Count	0	0	0	12	12
	% of Total	0.0%	0.0%	0.0%	10.0%	10.0%
Designer / Developer	Count	0	0	6	24	30
	% of Total	0.0%	0.0%	5.0%	20.0%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	0	0	6	24	30
	% of Total	0.0%	0.0%	5.0%	20.0%	25.0%
Business	Count	6	6	0	6	18

Analyst / Requirements Engineer	% of Total	5.0%	5.0%	0.0%	5.0%	15.0%
CIO / IT Director / Board Director	Count	0	6	0	6	12
	% of Total	0.0%	5.0%	0.0%	5.0%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count	0	0	6	12	18
	% of Total	0.0%	0.0%	5.0%	10.0%	15.0%
Total	Count	6	12	18	84	120
	% of Total	5.0%	10.0%	15.0%	70.0%	100.0 %

Respondents categorized by software configuration management activity

This question sought to find out whether software configuration management activity is necessary in ALM. On responding to the question, most of the experts agreed that the modeling and system design activity was very important in ALM. 70.0% of the respondents strongly agreed that it was very important while 15.0% agreed that it was important bringing to the total of 85.0% of those who agreed. 10% were not sure while 5% disagreed. With these results it shows that software configuration management activity is very key in ALM and thus must always be included.

4.2.9 Test management activity

Table XVI presents results basing on the recommendation of the different experts on the ALM Test management activity.

Table XVI.

		Test management activity is necessary in ALM			
		Not Sure	Agree	Strongly Agree	Total
Development Manager / Other IT Manager	Count	0	6	6	12
	% of Total	0.0%	5.0%	5.0%	10.0%
Designer / Developer	Count	0	0	30	30
	% of Total	0.0%	0.0%	25.0%	25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count	0	6	24	30
	% of Total	0.0%	5.0%	20.0%	25.0%
Business Analyst / Requirements Engineer	Count	6	0	12	18
	% of Total	5.0%	0.0%	10.0%	15.0%
CIO / IT	Count	0	0	12	12

Director Board Director	/% of Total	0.0%	0.0%	10.0%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count % of Total	0 0.0%	6 5.0%	12 10.0%	18 15.0%
Total	Count % of Total	6 5.0%	18 15.0%	96 80.0%	120 100.0 %

Respondents categorized by test management

From the results depicted in Table XVI, most of the experts agreed that the Test management module was very important in ALM. 70% of the respondents strongly agreed that it was very important while 15% agreed that it was important bringing to the total of 85% of those who agreed. The test management module is used to verify that the application complies with the requirements defined in the initial steps of the process (requirements management). It also ensures that the application meets the expectations of the users and all the other stakeholders that will need to support it throughout its lifecycle since testing is always advocated in all stages of SDLC which leads to development of applications requested by the users thus ensuring quality. It can also be noted that all the testers advocated for this module since it caters for all their testing needs. This results show that test management is also key in ALM.

It is also noted that more than 85% of all the staff with different designation advocate for Test management module since its functionality cuts across all designations but used more by Testers/ Test engineers as shown above since more than 80% of them recommend it.

4.2.10 Defect management activity

Table XVII presents results basing on the recommendation of the different experts on the ALM defect management activity.

This question sought to find out whether defect management activity is necessary in ALM. On responding to the question, most of the experts agreed that the modeling and system design activity was very important in ALM. 80.0% of the respondents strongly agreed that it was very important while 15.0% agreed that it was important bringing to the total of 95.0% of those who agreed while 5% were not sure. With these results it shows that software configuration management activity is very key in ALM and thus must always be included.

Table XVII.

		Defect management activity is necessary in ALM			
		Not Sure	Agree	Strongly Agree	Total
Development Manager Other IT Manager	Count /% of Total	0 0.0%	0 0.0%	12 10.0%	12 10.0%
Designer Developer	/Count % of Total	0 0.0%	6 5.0%	24 20.0%	30 25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count /% of Total	0 0.0%	0 0.0%	30 25.0%	30 25.0%
Business Analyst Requirements Engineer	Count /% of Total	0 0.0%	12 10.0%	6 5.0%	18 15.0%
CIO / IT Director Board Director	Count /% of Total	0 0.0%	0 0.0%	12 10.0%	12 10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count % of Total	6 5.0%	0 0.0%	12 10.0%	18 15.0%
Total	Count % of Total	6 5.0%	18 15.0%	96 80.0%	120 100.0 %

Respondents categorized by defect management

4.2.11 Release management activity

Table XVIII presents results basing on the recommendation of the different experts on the ALM release management activity.

From the results shown in Table 4.16, most of the experts agreed that the release management activity was very important in ALM. 60.0% of the respondents strongly agreed that it was very important while 35.0% agreed that it was important bringing to the total of 95.0% of those who agreed while 5.0% were not sure. This also depicts that the release management module is very important with more than 95% of the I.T experts agreeing to its importance since it functions to track release of applications to the production environment also referred to as deployment. Rolling out of an application always involves all parties thus its importance. Successful release of an application means success to a project. Any changes made during this stage always affect all players in the SDLC stages since they have to be traced back to requirements gathering. Different product type deployment have different attributes and specifications.

Table XVIII.

		Release management module is required in ALM			
		Not Sure	Agree	Strongly Agree	Total
Development Manager Other IT Manager	Count /% of Total	6 5.0%	0 0.0%	6 5.0%	12 10.0%
Designer Developer	/Count % of Total	0 0.0%	6 5.0%	24 20.0%	30 25.0%
Tester / Test Manager / Test Designer / Test Analyst / Analyst	Count /% of Total	0 0.0%	12 10.0%	18 15.0%	30 25.0%
Business Analyst Requirements Engineer	Count /% of Total	0 0.0%	12 10.0%	6 5.0%	18 15.0%
CIO / IT	Count	0	6	6	12

Director Board Director	/% of Total	0.0%	5.0%	5.0%	10.0%
Architect (Technical, Infrastructure, Security, Enterprise, etc.)	Count % of Total	0 0.0%	6 5.0%	12 10.0%	18 15.0%
Total	Count % of Total	6 5.0%	42 35.0%	72 60.0%	120 100.0%

Respondents categorized by the release management activity

V. FINDINGS, SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

A. Key Findings

According to the findings, ALM activities had a great effect on Software quality assurance. The constructs, namely Creation and management of lifecycle artifacts, Traceability, Reporting, Communication, Process support and Tool integration were also studied in this paper and several of them stood out clearly to aid in quality assurance of software developed.

1. ALM activities

This paper was set out to find the Application Lifecycle Management activities involved in software development and their effect on quality assurance. From the research, several activities were identified that contribute to ALM and quality assurance. These activities of ALM were defined by iteratively constructing and demonstrating the ALM framework based on literature and results presented in the paper. The results of the study contain four main elements as contributors of SQA that include requirement management, test management, Defect management and release management. These activities form the basis of the paper and are the main contributors to quality software development.

Requirement management is involved with ensuring the requirements are exhaustively captured and available across the entire software development lifecycle. With better communication among the tools this requirements should be traceable in all stages thus build on quality of software developed.

Test management is involved with ensuring that the software developed is tested across the entire software development lifecycle. Testing is a continuous process from inception of an idea to offering support to a developed software application. By this it means that linking software testing to the ALM activities could greatly improve on Software quality assurance.

Defect management is also another very important aspect to be covered in ALM activities since it deals with the management of errors and bugs found on the system. Thus with the integration of defect management on the ALM activities would greatly improve on the management of bugs and errors thus ensure an error free application is developed thus improving on software quality assurance.

Release management is the aspect of managing the process of rolling out a system to work environment 'Going live'. With the integration of this module in ALM means better and managed release of a system. With release management not taken into consideration even a well-developed system could fail thus the need to integrate this tool for better management of release which would lead to Software quality assurance.

CONCLUSION

This paper proposes a list of activities that could greatly improve on quality assurance if put into consideration in ALM. This paper studies the concept of ALM based on literature and case study performed on different middle level organizations. The main contribution of the paper is the ALM activities that contribute majorly to Software quality assurance and its demonstration relating to SME's systems.

The paper introduces four main ALM activities i.e. Requirement Management, Test Management, Defect Management and Release Management as being the main contributors of Software Quality Assurance in software development thus with the integration of this activities using tools could greatly improve on SQA.

The paper also found out that the attributes personnel experience and process support also had an effect on SQA. Personnel experience on the various ALM activities could easily assist in reducing errors created during management of artifacts of a given stage and thus have an effect on SQA. Process support also has an effect since already existing solutions could be applied to reduce on cost that could be incurred and since personnel are already used to existing tools then it would be easier for them to adopt and work with it with minimal errors thus improving on SQA.

The paper also proposes the use of subversion (SVN) to manage different artifacts found in different activities since it's an open source tool and can be used for version control to keep track of all activities involved in each stage of software development. With SVN in hand communications can be traced, software stages traced and changes concerning creation of artifacts, changes of artifacts and management of artifacts and Tests can be traced and managed easily and this could greatly lead to improved SQA.

In conclusion, the objective of this study was achieved as several activities were identified and the ones that stood out contributed to quality assurance in software development.

Recommendations and Further Research

The paper has been oriented in a horizontal way, i.e., by analyzing ALM as a whole, instead of with a vertical orientation, i.e., analyzing each ALM element in depth. ALM as a concept, however, is wide and each ALM element presented in the paper is worth its own research effort, for instance, traceability, tool integration or process support. The vertical in-depth research of each ALM element was therefore set outside the scope of this paper. The vertical study of elements is a potential topic for future research.

This research has limited the scope to the development cycle of the product even though the full lifecycle is broader, and it should therefore be the subject of future research. This means that research should be extended towards other lifecycle phases to cover also operation and maintenance phase of the lifecycle.

References

[1] Kääriäinen, J. E. (2009). Extending global tool integration environment towards lifecycle management. Retrieved 03 23, 2014, from http://www.springerlink.com:

http://www.springerlink.com/index/R285P4675535KU2P.pdf

[2] Carrillo, J. M. (2008). Application Lifecycle Management Meets Model-Driven Development. Retrieved 05 01, 2014, from http://drdobbs.com/architecture-and-design/210300020

[3] Georgi A. Markov, O. R. (2011). Towards an industrial ALM (Application Lifecycle) Tool Integration. Karlskrona: Blekinge Institute of Technology.

[4] Weiß, G. P. (2009). Software engineering – processes and tools. Workshop on Tool-Supported Requirements Management in Distributed Projects. Munich: Hagenberg Research.

[5] Doyle, C. (2007). The importance of ALM for aerospace and defence (A&D), Embedded. ESE Magazine.

[6] Schwaber, C. (2006). The Changing Face of Application Lifecycle Management. Forrester Research.

[7] Shaw, K. A. (2007). Application Lifecycle Management for the Enterprise. Retrieved May 01, 2014, from http://www. serena. com: http://www. serena. com/Docs/Repository/company/Serena_ALM_2. 0_For_t. pdf

[8] Goth, G. (2009). Agile Tool Market Growing with the Philosophy, IEEE Software.

[9] Kravchik, M. (2009). Application Lifecycle Management Environments. Past, Present and future.

[10] Moore, R. R. (2007). Scrum at a Fortune 500. AGILE 2007, 175 - 180.

[11] Medina-Domínguez, F. S.-S. (2007). Extending microsoft team foundation server architecture to support collaborative product patterns. International Conference on Software Process (ICSP 2007). Minneapolis, USA.

[12] Dearle, A. (2007). Software deployment, past, present and future, Future of Software. Minneapolis, MN, USA.

[13] Heindl, M. R. (2007). Requirements management infrastructures in global software development – towards application lifecycle management with role-based intime notification. International Conference on Global Software Engineering (ICGSE).

[14] Polit, D. F. (2004). Nursing research: Appraising evidence for nursing practice . Philadelphia: Wolters Klower/Lippincott Williams & Wilkins.

[15] Abramovici, M. (2007). Future Trends in Product Lifecycle Management (PLM). The Fu- ture of Product Development: Proceedings of the 17th CIRP Design Conference, (pp. 26–28, pp. 665–674). Berlin, Germany.

[16] Rossberg, J. (2008). Pro Visual Studio Team System. Application Lifecycle Management. Apress.

[17] Murta, L. W. (2010). The Configuration Management Role in Collaborative Software Engineering. Springer-Verlag, Berlin, Heidelberg.

[18] Gotel, O. a. (1994). An Analysis of the Requirements Traceability Prob- lem. Proceedings of the First International Conference on Requirements Engineering, (pp. 94–101).

[19] Sommerville, I. (2011). Software Engineering. Boston: pearson.

[20] Kotonya, G. A. (1998). Requirements Engineering: Process and Techniques. Chichester: John Wiley & Sons.

[21] Ramesh, B. a. (2001). Toward reference models for requirements traceability. IEEE Transactions on Software Engineering, (pp. 58–93).

[22] Toranzo, M. (1999). Multiview++ Environment: Requirements traceability from the perspective of different stakeholders. WER99 – II IberoAmerican Workshop on Requirements Engineering. Buenos Aires.

[23] Gills, M. (2005). Survey of Traceability Models in IT projects. roceedings of the ECMDA Traceability Workshop (ECMDA-TW), (pp. 39–46). Nuremberg, Germany.

[24] Dömges, R. A. (1998). Adapting traceability environments to project-specific needs. Communications of the ACM, 54–62.

[25] Espinoza, A. A. (2008). A Proposal for Defining a Set of Basic Items for Project-specific Traceability Methodologies. Proceedings of the 2008 32nd Annual IEEE Software Engineering Workshop, (pp. 175–184).

[26] Moreira, M. (2004). Software configuration management implementation roadmap. John Wiley & Sons.

[27] Estublier, J. (2000). Software Configuration Management. The Future of Software Engineering, 22nd International Conference on Software Engineering (ICSE 2000), (pp. 279–289).

[28] Swebok. (2004). Guide to the Software Engineering Body of Knowledge. IEEE Computer Society.

[29] Macfarlane, I. a. (1995). Requirements traceability in an integrated development environment. Proceedings of the Second IEEE International Symposium on Requirements Engineering, (pp. 116–123).

[30] Crnkovic, I. F. (1999). Processing requirements by software configuration management. Proceedings of the EUROMICRO'99 Conference, (pp. 260–265).

[31] Kolawa, A. (2006, January). The Future of ALM and CM, CM Journal. Retrieved 04 16, 2014, from http://www.cmcrossroads. com: http://www.cmcrossroads. com/cm-journal-articles/6651-the-future-of-alm-and-cm

[32] Yin, R. K. (2003). Case study research: Design and methods (3rd ed.). Thousand Oaks, CA: Sage.

[33] Yegidis, B. L. (2002). Research methods for social workers (5th ed.). Boston: Allyn & Bacon.

[34] Sandelowski, M. (2000). Combining Qualitative and Quantitative Sampling, Data Collection, and Analysis Techniques in Mixed-Method Studies. Retrieved May 01, 2014, from http://www.ryerson.ca: http://www.ryerson.ca/~mjoppe/rp.htm

[35] Ross, K. C. (2002). Sampling And Surveying Handbook. USA: Maxwell AFB.

[36] Mugenda, O. M. (2003). Research Methods – Quantitative & Qualitative Approaches. Bungoma: African Centre for Technology Studies.

[37] Santos, M. P. (2008). Collaborative Application Lifecycle Management with IBM Rational Products. An IBM Redbooks publication.