Analysis of Software Development Practices followed by In-House IT as Service Provider in Manufacturing Industry

Niketa Panchal, Dr. Akash Saxena
Research Scholar[1], Professor[2]
Rai University,Ahmedabad, India

Abstract: This paper emphasizes on software development practices adopted by in-house IT department in manufacturing industry. It is very difficult to get all the requirements in beginning of the project for such department. Clients perpetually arise new necessities after using software. They may raise additional functionality in existing product or ask to remove some of functionality. Software engineering caters such dynamic requirements using traditional software process models and agile methodologies. Software engineers and project managers decide strategy to follow relevant software process model for project. They have challenge to follow all standard phases mentioned in software process model due to limited resources available. Consumers expectations are user friendly software product on or before time rather product with technical documents and prototypes, such scenario inspire software developers to sidestep such elements. This situation may result in lack of documentation, inadequate testing, increase in maintenance. The aim of this analysis is to find gaps in working practices of in-house IT department as software service provider.

Index Terms **- Manufacturing industry, functionality, team size, line of code, Analysis, Development Duration, Dynamic, maintenance, agile, traditional.**

I. INTRODUCTION

IT industries has played significant role to make ease of business in all the sectors, be it banking, industries, education, transportation or our daily life. It is very important that any software developed enhances productivity for users. There are many software developments models has been evolved over the years to serve software developers which make their business easy and help them to build powerful, robust, user friendly quality software. These models are broadly categorized as Traditional models and Agile models. Both models have their own pros and cons. If same software is developed using both agile and traditional method, it would be difficult to integrate different objects [2]. Agile methodologies are widely used in many engineering organizations to reduce their reversal times, decrease misused efforts and rise profitability [1]. Depending upon the project requirement and resource availability, software developers choose model suit their project. We can divide software development teams in two main categories one is software engineers working for software company as production department and another is software engineers are working as service provider in non-IT industries. This paper focuses on later category in which they have very tough condition to work with limited resources and deliver quality product on time which help to company production to enhance productivity.

Iteratively developed software meets user requirements for small objects and takes go ahead approval for next in agile methodology. This approach ensures quick delivery by testing each small object and constant integration [8]. Self-forming cross-functional teams works on software requirements to provide solutions for evolving requirements using agile approach with moral to deliver cost effective and quality software. [9]. Incessant Changes from consumers due to several parameters like technology, business requirement, etc. can be addressed using Agile approach for business enterprise. [10]. In person communications and teamwork is the fundamental of agile methodology using incremental method. Correct testing confirm increments are error free and meets customer requirements. [11].

L&T Hydrocarbon Engineering (LTHE), a wholly owned subsidiary of Larsen & Toubro Limited (L&T), serves the Oil and Gas sector around the world. Organized under Offshore, Onshore, Construction Services, Modular Fabrication and Engineering Services verticals, the company delivers 'design to build' engineering and construction solutions across the hydrocarbon spectrum [7]. The core business of Modular Fabrication Facility (MFF) is to manufacture Offshore platform which involves Engineering, Procurement, Construction (EPC) activities. These activities are supported by many IT systems and software including Engineering software, SAP and eALPS. eALPS is in-house developed software consist of lots of web applications integrated with SAP and engineering software which is developed and maintained by MFF-IT team. The software focuses to support all manufacturing activities carried out at yard to meet user's expectations with on time delivery and quality. A small application to begin with evolved as large software today by continuous developments as per user requirements. The aim is to study their way of working, which methodology they are following, which software development methodology they are using, which difficulties they are facing, how much size of each software, how much time they took to develop software and how much time they spend in

maintenance, what are the type of maintenance they are doing and what is the amount of it. This paper's prime objective is to find out gaps in software development process followed by this team.

Software Development Team:

The team comprises of team leaders, Senior developer, junior developer which ensures to get in depth insight of their user requirements, analyze requirements and participate in software development starting from requirement gathering to maintenance. The team strongly believes that their clients (users) are very important member of the team. They are with the IT team most of the time during requirement gathering, testing and release phase.

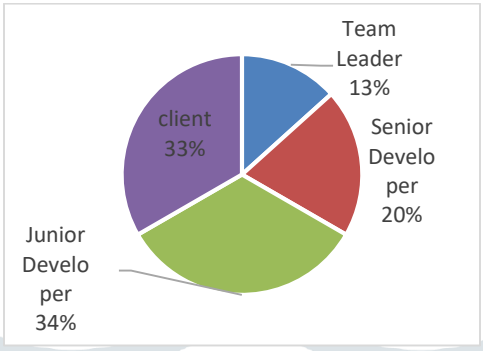

Figure-1: Software Development Team distribution

Line of Codes (LOC) analysis:

The same IT team is working on all LOC category of projects from variety of domains, depending on requirement client team varies. The challenging and interesting part of this scenario is that same team has to deliver this variety of products in terms of LOC and domain hence there are instances when they have to work simultaneously on multiple projects.

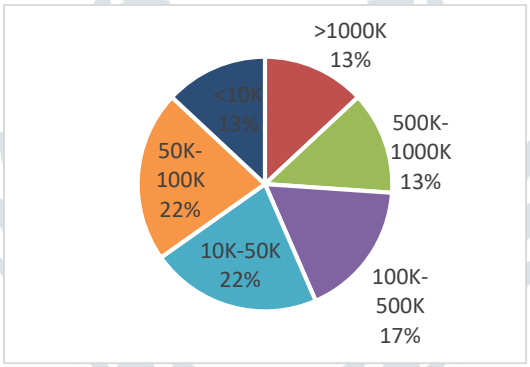

Figure-2: Line of Code wise software distribution

Each domain is having different LOC size of software. Manufacturing is core business of industry so large software users are from manufacturing. Quality Assurance (QA), Quality Control (QC), Non-destructive Testing (NDT), Third party authority(TPA) and Safety is another important factor in manufacturing industry so we can see that software having client as QA, QC , NDT, safety also considerably bigger in size.

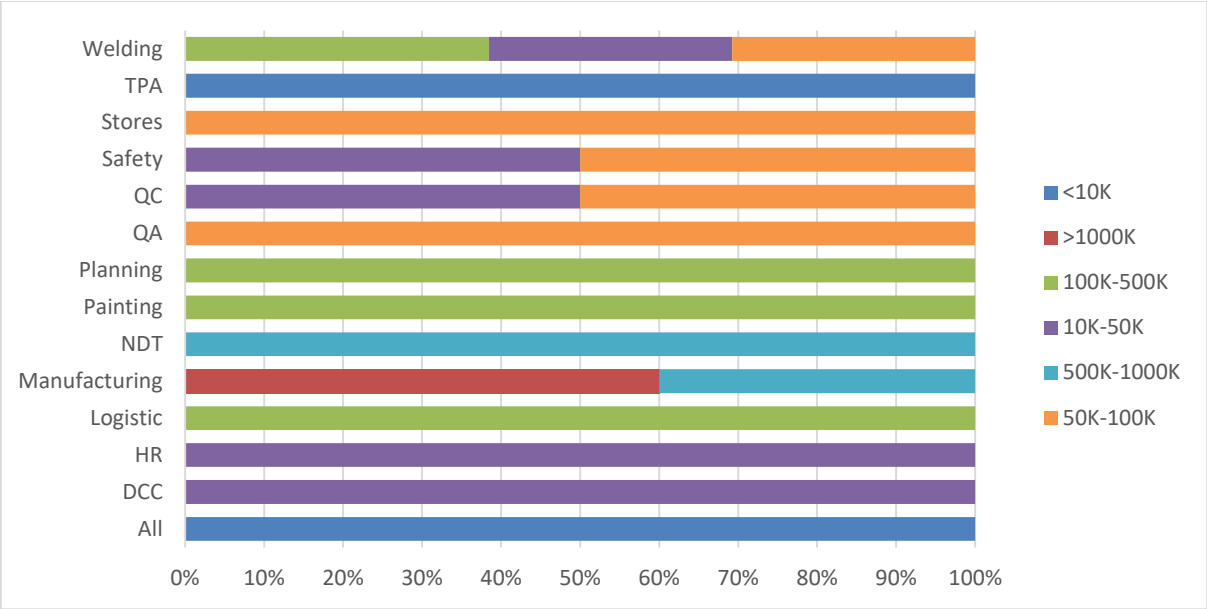

Figure-3: Software domains LOC wise distribution

Development Time Duration:

Another aspect of study is that various domains projects are completed in how much duration. This software are rages from small software application to large EPR system. Some of them are completed in few months and some took years to complete.

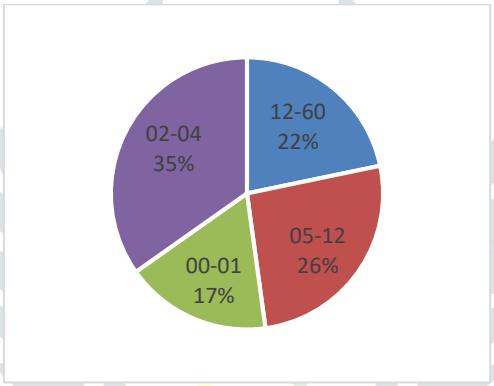

Figure-4: Completion duration wise software distribution

The level of project criticality for business is considered as high, low and medium. The analysis in below chart shows that most high critical projects are either long duration projects or medium duration projects and medium duration projects' criticality is medium.

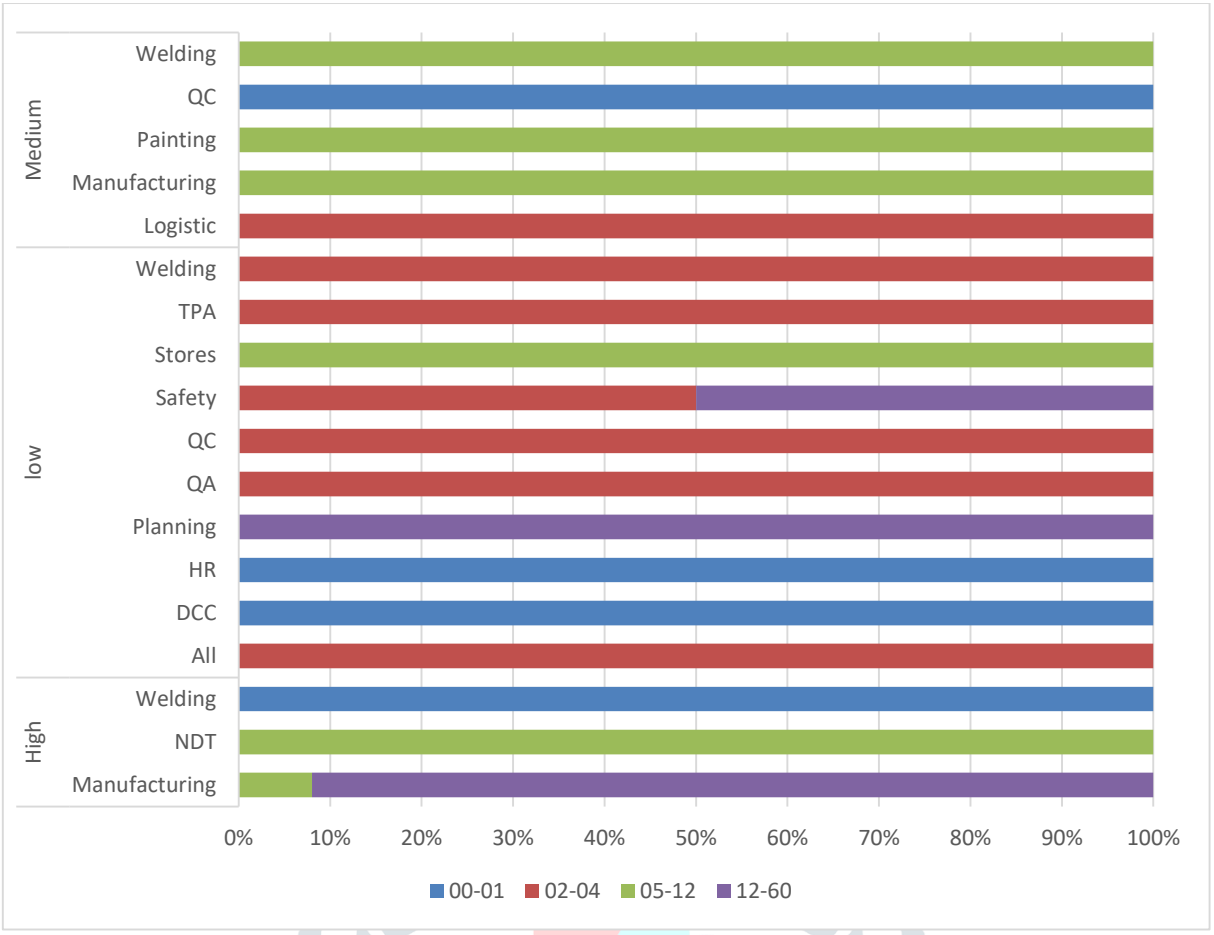

Figure-5: Completion duration (months) for project criticality and domains

The challenge is that software team must deliver these critical projects on before time with small team size. There is high probability that delay in software delivery impact company business. Another below analysis indicates that business critical projects are having 10K LOC to >100K LOC. The normal trend is that a greater number of LOC takes more project completion time and a smaller number of LOC takes lesser project completion time. However, there are exceptions that small numbers of LOC take more project completion time and vice versa. This means that this software team needs to be very dynamic to deliver the project.

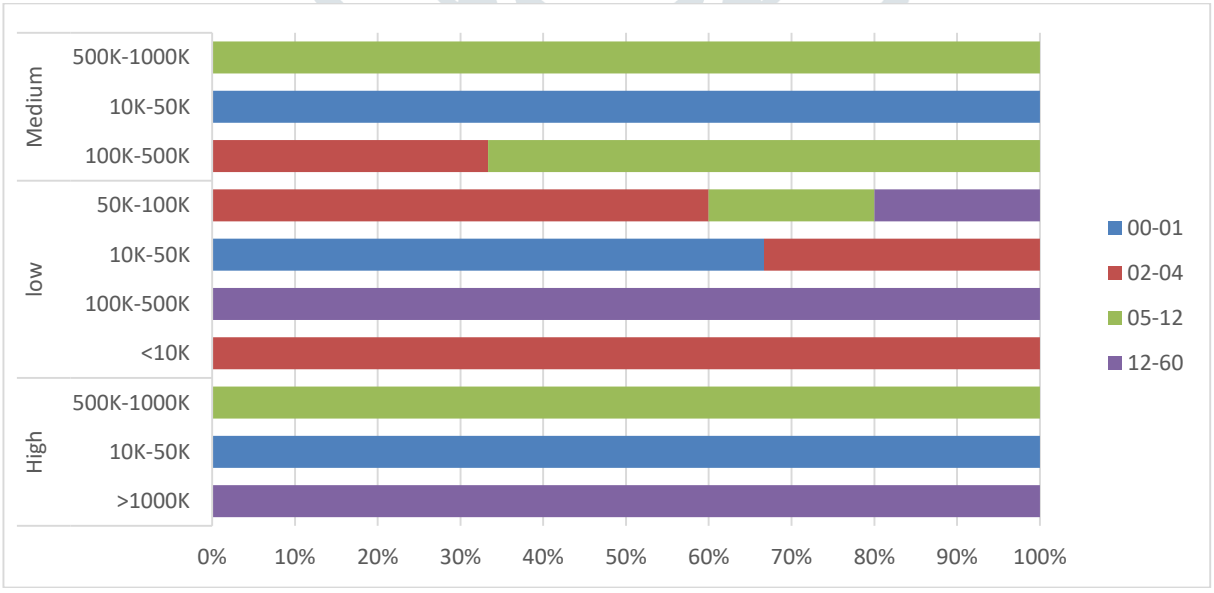

Figure-6: LOC duration (months) for project criticality

Software Maintenance:

Below chart describes maintenance type wise distribution for software. Large percentage of perfective maintenance indicates that users have come up with lots of new requirements to update existing software. Next major contributor is corrective maintenance

which indicates that there are number of instances where software was released with errors. Low preventive maintenance indicates that there is scope of performance improvement in the software. Low percent of adaptive maintenance is due to update in operating system, framework or update in database application.

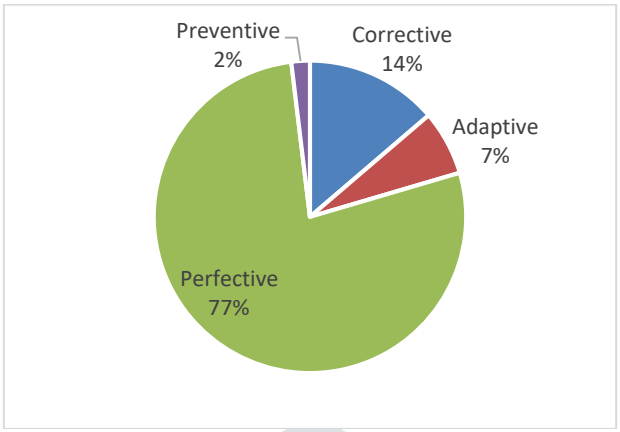

Figure-7: Maintenance type distribution

Below analysis shows that high critical and large project has undergone lots of maintenance. Majority of this maintenance is perfective maintenance. The main reason is that users ask to release initial working functionalities, they use these functionalities and derive another requirement to enhance or top up used functionality. Many of times it is due to company clients come up with new requirements. It indicates that perfective maintenance is routine scenario for such industries.

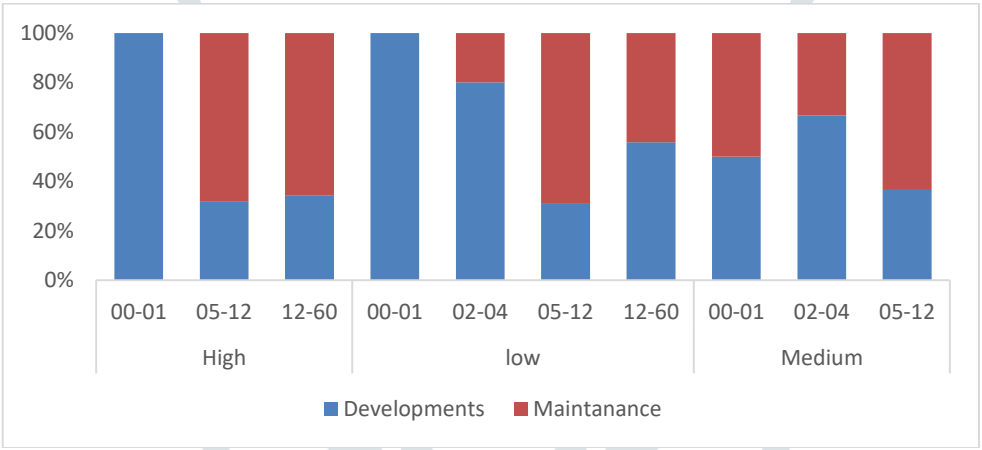

Figure-8: Software development v/s maintenance ratio for project criticality

Corrective maintenance is one is the important parameter for software team. Below chart shows corrective maintenance breakup for different duration projects. Corrective maintenance is tending to zero for small duration projects indicates that methodology adopted by this team to develop short term project is not creating much problems in terms of corrective maintenance. Corrective maintenance should tend to zero for any good software. High percentage of corrective maintenance for long and medium duration projects indicates that there is some gap in their working practice.

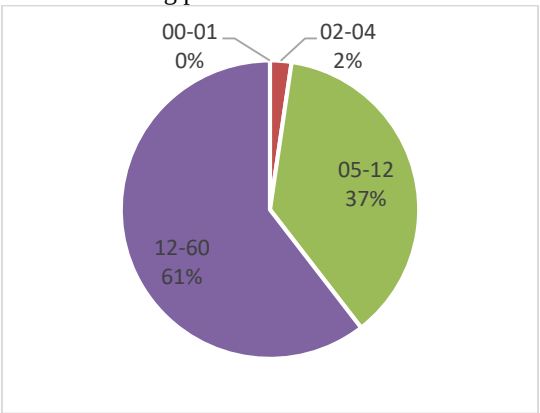

Figure-9: Corrective maintenance distribution for different duration (months) project

Below chart shows perfective maintenance breakup for different duration projects More perfective maintenance indicates that either users are not having vision to give full requirement to software team or some external parameters make users come up with

additional requirements for medium to large duration projects. It is normal trend in manufacturing industry where different clients come up with advanced requirements which results in high percentage of perfective maintenance.

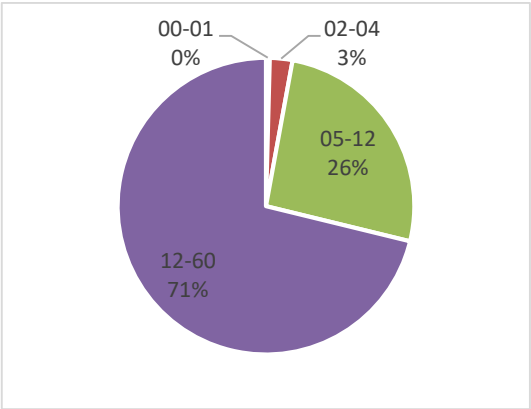

Figure-10: Perfective maintenance distribution for different duration (months) project

A typical software development trend of long duration software is indicated in below chart. The chart shows that new developments finishes in in initial period and software gets released. The amount of iteration is less during new development cycle. After using the software, users initiated additional requirements which is considered as maintenance phase of software. The chart shows that iterations are more with during maintenance phase than in new development phase

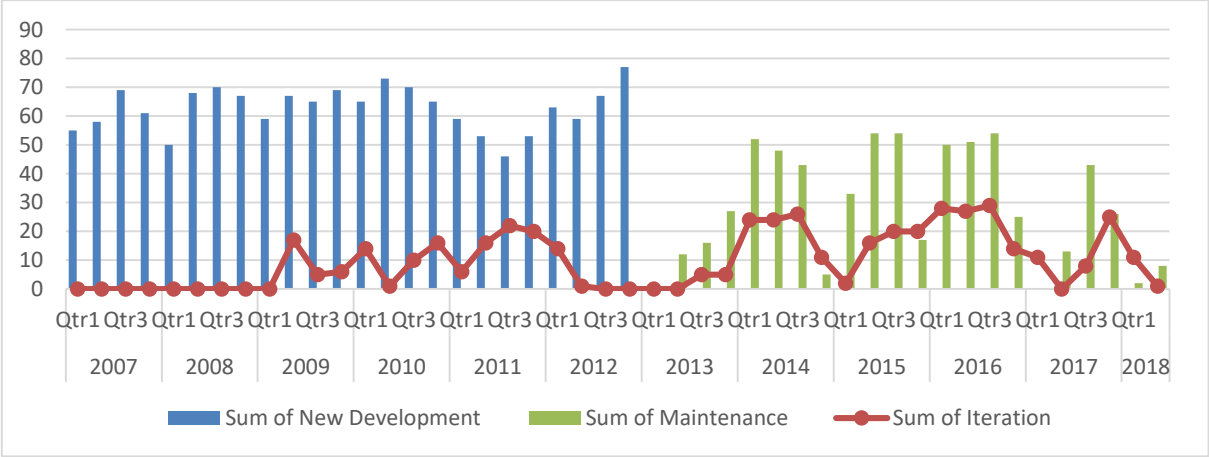

Figure-11: Large software development trend

Software process model stages:

Below chart shows that standard SDLC stages are followed in software development. High percent of projects are released without proper testing. This may lead to increase in corrective maintenance percentage.

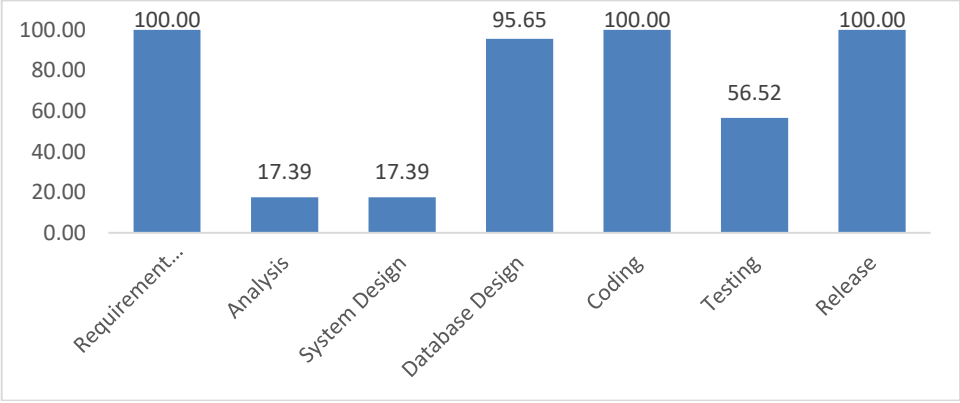

Figure-12: SDLC stage implementation

Below analysis shows that major time is distributed among three stages Requirement gathering, Database Design and Coding. Focusing on these stages is the main reason why the team is able to deliver projects as per expectation. They are having experienced software engineers who have accumulated good amount of business domain knowledge which helped developers to develop more efficient software.

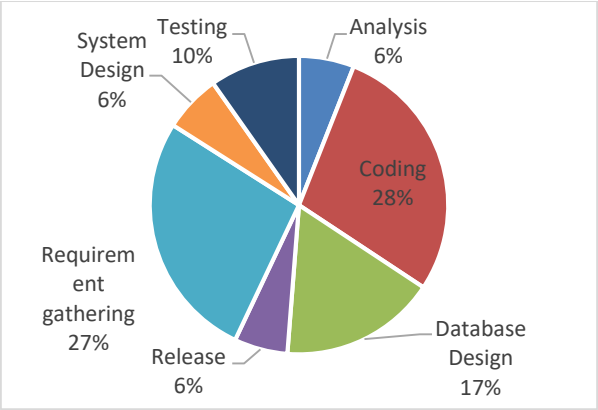

Figure-13: SDLC stage time distribution

Below chart indicates that almost one fourth times is utilized in Requirement gathering stage for all duration type of projects, Good amount of time is invested in database design, Coding which is normal in any software development cycle. Things to note down is that Analysis, System design and Testing phases are not implemented in small duration projects. All stages are implemented in long duration projects.

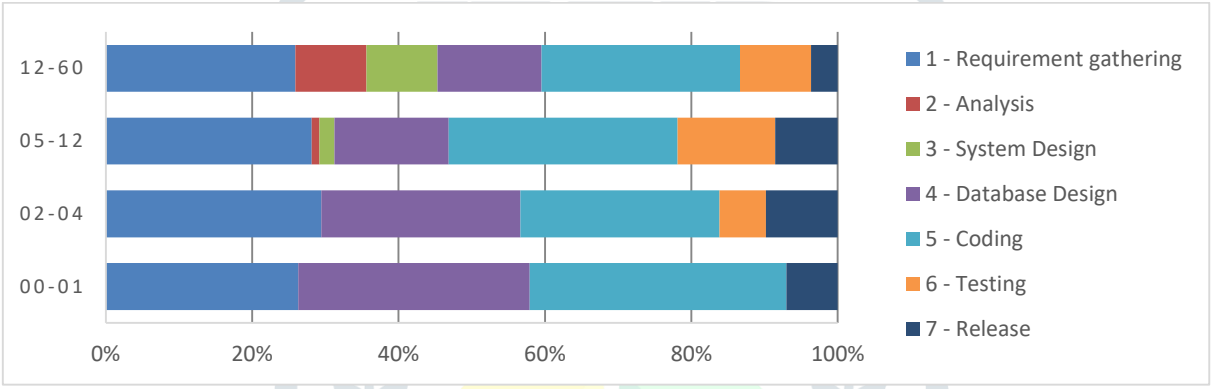

Figure-14: SDLC phase distribution for different duration (months) project

Analysis shows that documentation is very poor although software are delivered on time with quality. This is the major area of improvement required for this team. This improvement should not come at project completion duration cost and high increase in manpower.

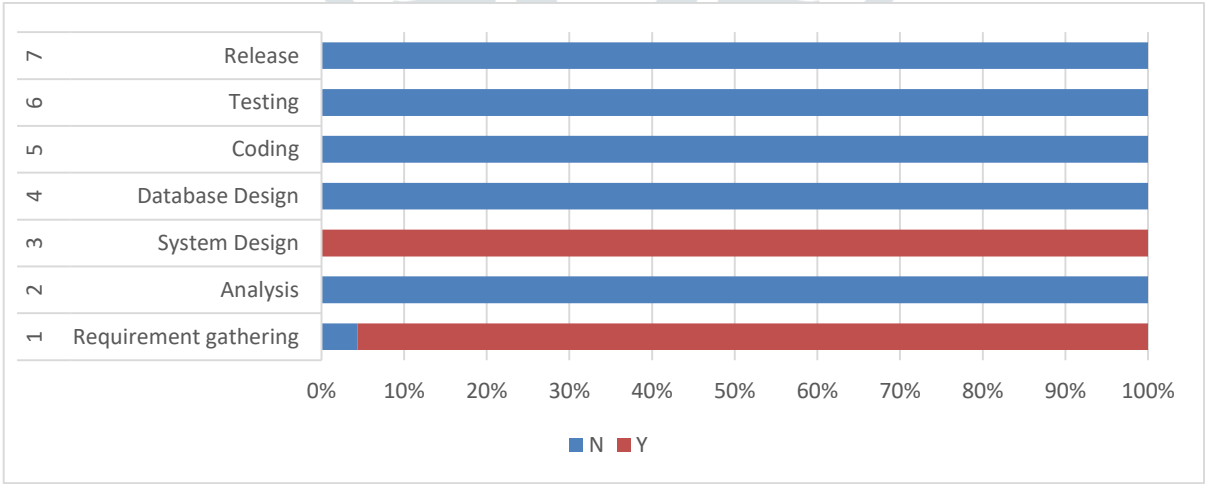

Figure-15: Documentation in SDLC stages

Conclusion:

IT team at manufacturing environment working as a software service provider is having small team size to serve different domain software. Most of the time they work on multiple projects simultaneously. This parallel work may be in combination of two or more new projects, two or more old project maintenance, one old project maintenance and one new project. Being an in-house service provider, team focuses on on-time quality software delivery rather than follow standard IT practices. It consists of poor

documentation, some functionalities release without testing. Perfective maintenance percentages are significantly high for long duration projects and these projects are critical for business. Corrective maintenance percentage is noteworthy. Preventive maintenance percentage is very low, if is increased software will perform more smoothly. More time invested in requirement gathering, database design, coding is the key behind the success of this team. Survey data shows that traditional process model is followed in some cases and agile methodology is followed in some cases. This IT team is not following any specific software process model, but they are utilizing good practices from both traditional and agile methods and delivering quality software products.

REFERENCES

[1] Naresh Mehta, J. G. (2017, may). Agile in Multisite Software Engineering.

[2] Barry Boehm, R. T. (2005). Management challenges to implementing Agile Process in Traditional Development organizations.

[3] Kiran Hiwarkar, A. D. (2016, march). "Comparative Analysis of Agile software Development Methodologies- A Review"

[4] Yu Beng Lean, W. K. (2012). "Software Development life cycle Agile V/S Traditional Approaches". *IPCST*.

[5] Sanjay Misra, L. F. (2014). "A Brief Overview of software process model: Benefits, Limitations and Application in Practice".

[6] Rantanen, E. (2017, April). "Requirements Engineering in Agile Software Projects".

[7] http://www.lnthydrocarbon.com/

[8] Deniz Iren, M. D. (2011). Usability studies in agile software development.

[9] K.C.Joshi, R. R. (2018, february). Integrating Security in agile software development.

[10] Richard Devor, R. G. (1997). Agile Manufacturing research: accomplishements and Opportunities. *IIE Transaction, US*.

[11] Rizwan Khan, D. P. (2016, january). Agile Approch for software Testing Process. Research Gate.