Awareness Regarding Importance of SDLC among IT Professionals and Students: A Survey

Syeda Nazia Ashraf
Lecturer
Sindh Madressatul Islam University,
Karachi, Pakistan

ABSTRACT
This paper aims to create awareness regarding importance and need of Software Development Life Cycle (SDLC) among IT Professionals and Students in executing projects. The SDLC itself is the solution for the success of projects but there is always a room for improvement and its awareness will highlight the areas of SDLC in detail and its need and importance. In this paper, we focus on strengths and weaknesses of SDLC. What are the impacts for selecting right SDLC process models? What are the Risk Factors involved in SDLC Phases? In this regard, some survey results also included to emphasis the importance of SDLC in IT based projects. Survey indicate that, in every organization SDLC implemented by different professionals of every field as well as students in their respective area of projects.

Keywords
SDLC, RAD, Risk Factor, RD, AD

1. INTRODUCTION
Software development life cycle (SDLC) is an important part of any software application development. Software Development Life Cycle (SDLC) is used in project management to develop (or alter existing) information systems or applications. The SDLC process provides Information Technology (IT) project managers with the tools to help guarantee effective implementation of systems or applications that satisfy strategic and business objectives. SDLC documentation provides a strategy to ensure that administrative leadership, functional managers, and users sign-off on the requirements and implementation of the system. The process provides management with the ability to design, develop, and implement a proposed system which is efficient and high-quality software that achieve or exceeds customer expectations, reaches completion within times, delivered on time and within a budget[1]. SDLC outlines the important stages of development life cycle which are used by analyst, system designers and developers to plan and execute sequence of activities required to achieve a quality software or system at scheduled time and estimated cost. This process involves contribution of client, analysts, designers, developers and end-users as per the existing stage of SDLC. A Systems Development Life Cycle (SDLC) contain six important phases that are essential for developers, such as requirement gathering and analysis, design, implementation or coding, testing, deployment and maintenance[2]. These phases create a hardware system only, a software system only or a combination of both to achieve or exceed customer's expectations. A number of software development methodologies defined and designed which are used during development process of software, these approaches are called "Software Development Process Models" e.g. Waterfall model, Incremental model, V-model, Iterative model, Rapid Application Development (RAD) model, Agile model, Spiral model, Prototype model etc. Each process model follows a specific life cycle to certify success in process of software development [1]. Several models are combined into some sort of hybrid methodology. Documentation is vital irrespective of the type of model chosen or planned for any application, and is usually done in parallel with the development process. Some methods work better for specific types of projects, but in the final analysis, the most important factor for the success of a project may be how closely the plan was followed[3].

2. IMPORTANCE OF SDLC
The aim of SDLC process is to produce a cost-efficient, effective and high quality product. Software Development Life Cycle (SDLC) is observed by different website and software developers all over the world. Need to implement this type of planning because SDLC could help us answer specific needs of different users. We need to have a carefully mapped SDLC. It is necessary of having a properly implemented SDLC model in creating a software or an online service. SDLC model is the key to Project Success.

2.1 Impact on developing programs without using SDLC Model.
Absence of proper documentation is one of the causes why a program without SDLC model is undesirable. There is no problem in the implementation of software but with no SDLC, there are no documentations to support the development of the program. Programmers could create their own documentation, but this is only for the program and nothing else. Development and troubleshooting are parts of proper documentation which is outlined in SDLC. A program that has no SDLC cannot be reproduced if there is a documentation that identifies the answer. Another disadvantage of developers who don't follow SDLC is the incapability to handle programs with complex needs. In-house developers may be able to force themselves to create programs but the larger picture is missed. In a business world, developers must work with business managers and there are certain needs that should be properly outlined.

2.2 Impact on Programs/Websites built using SDLC Model.
Through SDLC Model, developers will have a clear idea on what should be or should not be built. Since they already have an idea on the problems that should be answered, a comprehensive plan could be developed following a certain SDLC model. SDLC model enable developers to create a program that will answer different problems at the same time. Since everything will be arranged before a single code is written, the objective is clear and could be executed on time. Although there is a great chance of deviation from the plan, a good project manager will take care of that concern.

By applying SDLC Model, programs generate will have a clear documentation of development, structure and even coding. If there are problems once the program is adopted for end user, developers will always have the documentation to refer to when they need to look for any ambiguities. Instead of testing it repetitively which will halt the implementation for a while, developers will just explore the documentation and perform proper maintenance program. This means SDLC will revive, energize and strengthen the program. Instead of annoying developers in estimation if something goes wrong, SDLC will guarantee that everything goes effortlessly. It will also be a tool for maintenance, confirming the program developed will exist for a long time.

Although SDLC modeled program takes longer life time, every created program could serve be used in another program. This ability to adapt to different programs is ensured through proper documentation. Since each step is properly documented, some parts could be used to create another program. An ideal SDLC model will have the proper documentation of every step and every part of the program that there is a possibility of using the existing structures for another program. This will reduce the period of development which is good for software development. All this will lead to an improved and more efficient program. Since everything is arranged properly, developers will have more time to inspect the program for possible flaws.

3. STRENGTHS AND WEAKNESSES OF SDLC

Limited people in the modern computing world would use a strict waterfall model for their SDLC as many modern methodologies have outdated this thinking. Some will claim that the SDLC no longer applies to models like Agile computing, but it is still a term widely in use in Technology circles. The SDLC practice has advantages in traditional models of software development that lends itself more to a structured environment. The disadvantages to using the SDLC methodology is when there is need for iterative development or (i.e. web development or e-commerce) where stakeholders need to review on a regular basis the software being designed. Instead of observing SDLC from a strength or weakness perspective, it is far more important to take the best practices from the SDLC model and apply it to whatever may be most suitable for the software being designed.

A comparison of the strengths and weaknesses of SDLC:

Table 1: Strengths and weaknesses of SDLC

	Strengths	Weaknesses
1.	Formal review at the end of each phase allows maximum management control.	Major problem - end-user does not see the solution until the system is nearly complete. Users get a system that meets the need as understood by the developers; this may not be what was really needed, which results increased development time.
2.	This approach creates considerable system documentation.	Documentation is expensive and time-consuming to create. It is also difficult to keep current. As a result, it's hard to estimate costs, project overruns.
3.	Formal documentation ensures that system requirements can be traced back to stated business needs.	Often, users' needs go unstated or are misunderstood.
4.	It produces many intermediate products that can be reviewed to see whether they meet the user's needs and conform to standards.	Users cannot easily review intermediate products and evaluate whether a particular product (e.g., data flow diagram) meets their business requirements.
5.	Monitor large projects.	Increased development cost.
6.	Detailed steps.	Systems must be defined up front.
7.	Evaluate costs and completion targets.	Rigidity and stiff implementation instead of creativity. There are requirements that must be met and that is all that developers complete.
8.	Well defined user input.	User input is sometimes limited.
9.	Ease of maintenance.	A change in one phase will result in a later stage.
10.	Development and design standards.	Eliminating one or more steps for faster manufacturing process, will result system failure.
11.	Tolerates changes in MIS staffing	-

Although both sides have been weighed up here, it is obvious that the strengths are greater than the weaknesses[4].

4. IMPACT OF SELECTING RIGHT SDLC MODEL

In the software industry, a large number of projects fail and billions of dollars are spent on failed software projects. Poor selection process of software development life cycle (SDLC) models is one of the top reason of such failure. By selecting right software process model a better and high quality product can be found within budget and time. Many approaches are used for selecting a process model. Selection of an appropriate SDLC model based on different project characteristic categories are of the approach to discuss here. A comparison approach of SDLC process is presented, which is based on project characteristic categories and then categories are classified. Tables of SDLC models are compared, and give better selection process of SDLC models.

Comparison based on three project characteristic categories:

4.1 Project Team

4.2 User Community

4.3 Project type and Risk

4.1 Project Team

Whenever possible, it is necessary to select the individuals for the project team before selecting any SDLC process model. The characteristics of the team Table 1 are important in the selection process they are responsible for successful completion of the cycle, and they can assist in selection process. (Weightage from 1 to 9 which shows low degree to high degree) Characteristics of the project team members: -

1. New to problem domain: - the majority of team members are new to the problem domain for the project.
2. New to the technology domain: - the majority of the team members are new to the technology domain for the project
3. New to tools to be used: - the majority of team member are new to the tools to be used on the project
4. Any training available: - there is training available for the project team, if required
5. Comfortable with structure: - the team is more comfortable with structure than flexibility
6. Closely track by manager: - the project manager will closely track the team's progress

Table 2: Comparison based on Project Team

Project Team				
	Waterfall	**Spiral**	**RAD**	**Incremental**
New to problem domain	1	9	1	3
New to the technology domain	8	9	1	8
New to tools to be used	7	8	1	2
Any training available	2	1	8	9
Comfortable with structure	8	1	2	9
Closely track by manager	8	9	2	8

4.2 User Community

The early project phase can provide a good understanding of the user the User Community Table 2 and expected relationship with the project team for duration of the project. This understanding will assist you in selecting the appropriate model because some models are dependent on higher user involvement and understanding the project.

Characteristics of the user community: -

1. Availability of user representative restricted or limited: - the availability of the user reprehensive will be restricted or limited during the life cycle
2. User representative new to the system definition: - the user representatives are new to the system definition
3. User representative expert in problem domain: - are the user representatives are expert in problem domain
4. User representative want involve in SDLC: - the user wants to involve in all phases of the life cycle
5. User representative want to track project progress: - customer want to track project progress

4.3 Project Type and Risk

Examine the type of project and identified risk Table 3 in the planning phase. Some models are designed to accommodate highrisk management, while others are not. The selection of a model that accommodates risk management does not mean that you do not have to create an action plan to minimize the risk identified. The model simply provides a framework within which this action plan can be discussed and executed.

Characteristics of project type and risk: -

1. Integration project: - the project is a system integration project
2. Enhancement to an existing system: - the project is an enhancement to an existing available project
3. The funding for project: - the funding for the project is expected to be stable through-out the life cycle
4. Project reliability: - the project high reliability is must

Table 3: Comparison based on User Community

User Community				
	Waterfall	**Spiral**	**RAD**	**Incremental**
Availability of user representative restricted or limited	9	2	1	7
User representative new to the system definition	1	9	1	8
User representative expert in problem domain	1	2	9	8
User representative want involve in SDLC	1	1	9	2
User representative want to track project progress	1	9	2	1

Table 4: Comparison based on Project Type and Risk

Project Type and Risk				
	Waterfall	**Spiral**	**RAD**	**Incremental**
System Integration project	1	8	7	9
Enhancement to an existing project	1	2	9	8
Funding is stable throughout SDLC	9	2	8	1
High reliability must	1	9	2	8

If during the project something changes that causes the team to apply a different model that may be more appropriate; then the model can be changed during the execution of the project; but it should be done with careful consideration to the impacts of the project. Lastly, it is better to change the model than to use one that is not well suited to meet the needs of the project[5].

Based on observation, comparison, survey and experience Tables 5, 6, 7 are prepare and the steps in best life cycle selection are these:

1. Being familiar with the various models.
2. Review and analyze the types of work performed like development, enhancement, and maintenance.
3. Review the life cycle approach to standards required for your organization, your customer, or the type of project- ISO, IEEE, and so on.
4. Identify a set of phase and phase activates.
5. Evaluate the effectiveness of the life cycle framework, and implement improvements where needed[5].

Table 5: Suggested Model based on Team Property

S.No.	Project Team members	Suggested Model
1.	New to problem domain	Spiral
2.	New to the technology domain	Spiral
3.	New to tools to be used	Spiral
4.	Any training available	Incremental
5.	Comfortable with structure	Waterfall
6.	Closely track by manager	Spiral

Table 6: Suggested Model based on User Community

S.No.	User Communicate	Suggested Model
1.	Availability of user representative restricted or limited	Waterfall
2.	User representative new to the system definition	Spiral
3.	User representative expert in problem domain	RAD
4.	User representative want involve in SDLC	RAD
5.	User representative want to track project progress	Spiral

Table 7: Suggested Model based on Project Type and Risk

S.No.	User Communicate	Suggested Model
1.	System Integration project	Incremental
2.	Enhancement to an existing project	RAD
3.	Funding is stable throughout SDLC	Waterfall
4.	High reliability must	Spiral

5. RISK FACTORS INVOLVED IN SDLC PHASES

Each phase of the Software Development Life Cycle (SDLC) is susceptible to different types of risk factors. To deal with these risks properly, an acceptable understanding of the software development process issues, risks and their causes are necessary. Hence, the first step in handling these risks is to identify them.

5.1 Risk Identification Framework:

Risk Identification Framework is composed into five major subsections. Each one describes each phase in the SDLC with their activities and possible risk factors.

5.1.1. Requirements analysis and definition phase

The requirements analysis and definition phase comprises of many activities: Feasibility study, requirements elicitation, requirements analysis, requirements validation and requirements documentation.

5.1.1.1 Feasibility Study Activity

Risk factors for the feasibility study activity are inadequate estimation of project time, cost, scope and other resources, unrealistic Schedule and Budget, unclear Project Scope and insufficient resources.

5.1.1.2 Requirements Elicitation Activity

Risk factors for the requirements elicitation activity are unclear requirements, incomplete requirements, inaccurate requirements, ignoring the nonfunctional requirements, conflicting user requirements, unclear description of the real environment, and Gold Plating

5.1.1.3 Requirements Analysis Activity

Risk factors for requirements analysis activity are non-verifiable requirements, infeasible requirements, inconsistent requirements, non-traceable requirements, and unrealistic requirements

5.1.1.4 Requirements Validation Activity

Risk factors for the requirements validation activity are misunderstood domain-specific terminology, inaccurately expressing user requirements in natural language

5.1.1.5 Requirements Documentation Activity

Risk factors for the requirements documentation activity are inconsistent requirements data and RD (Requirements Document), non-modifiable Requirement Document

5.1.2. Design phase

This phase involves several activities: examining the requirements document (RD), choosing the architectural design method, choosing the programming language, constructing the physical model, verifying, specifying, and documenting design activities.

5.1.2.1 Examining the Requirements Document (RD)
Activity

Risk factor for examining the requirements document activity is: RD is not clear for developers

5.1.2.2 Choosing the Architectural Design Method Activity

Risk factor for choosing the architectural design method activity is:

Improper AD (Architectural Design) method choice

5.1.2.3 Choosing the Programming Language Activity

Risk factor for choosing the programming language activity is: Improper choice of the PL

5.1.2.4 Constructing the Physical Model Activity

Risk factors for constructing the physical method activity are too much complex system, complicated design, large size components, unavailable expertise for reusability, less reusable components than expected

5.1.2.5 Verifying Design Activity

Risk factors for verifying design activity are difficulties in verifying design to requirements, many feasible solutions, and incorrect design

5.1.2.6 Specifying Design Activity

Risk factors for specifying design Activity are difficulties in allocating functions to components, extensive specification, omitting data processing functions, large amount of tramp data

5.1.2.7 Documenting Design Activity

Risk factors for documenting design activity are incomplete design document, large design document, unclear design document, and inconsistent design document

5.1.3. Implementation and Unit Testing Phase

This phase integrates two main activities; coding and units testing in an iterative manner

5.1.3.1 Coding Activity

Risk factors for the coding activity are non-readable design document, programmers cannot work independently, developing the wrong user functions and properties, developing the wrong user interface, PL does not support architectural design, modules are developed by different programmers, complex, ambiguous, inconsistent code, different versions for the same component, developing components from scratch, large amount of repetitive code, inexperienced programmers, too many syntax errors,

technology change

5.1.3.2 Unit Testing Activity

Risk factors for the unit testing activity are high fault rate in newly designed components, code is not understandable by reviewers, lack of complete automated testing tools, testing is monotonous, boring and repetitive, informal and ill-understood testing process, not all faults are discovered in unit testing, poor documentation of test cases, data needed by modules other than the under testing one, coding drivers and stubs, poor Regression Testing

5.1.4. Integration and System Testing Phase

This phase integrates three activities: integration, integration testing, and system testing activity.

5.1.4.1 Integration Activity

Risk factors for the integration activity are difficulties *in* ordering components' integration, integrate the wrong version of components, omissions or oversights

5.1.4.2 Integration Testing Activity

Risk factors for the integration testing activity are a lot of bugs emerged during the integration, data loss across an interface, integration may not produce the desired functionality, difficulties in localizing errors, and difficulties in repairing errors.

5.1.4.3 System Testing Activity

Risk factors for the system testing activity are unqualified testing team, limited testing resources, inability to test in the operational environment, impossible complete testing, testers rely on process myths, testing cannot cope with requirements change, wasting time in building testing tools, the system being tested is not testable enough.

5.1.5. Operation and Maintenance Phase

Installation, operation, acceptance testing, and maintenance are the activities in this phase.

5.1.5.1 Installation Activity

Risk factors for the installation activity are problems in installation, the effect on the environment, and change in environment

5.1.5.2 Operation Activity

Risk factors for the operation activity are new requirements emerge, Difficulties in using the system

5.1.5.3 Acceptance Testing Activity

Risk factors for the acceptance testing activity are user resistance to change, missing capabilities, too many software faults, testers do not perform well, suspension and resumption problems, insufficient data handling

5.1.5.4 Maintenance Activity

Risk factors for maintenance activity are the software engineer cannot reproduce the problem, problems in maintainability, budget contention

5.1.5.5 Risk Factors common to all SDLC phases

Some risks factors intimidate all the phases of the SDLC, starting from the initial inspection of the project to the final release. The risk factors that are common to all SDLC phases are continually changing requirements, time contention, project funding loss, team turnover, data loss, and miscommunication[6].

6. METHODOLOGY

A survey has conducted online and collect responses from different IT Professionals and students from IT field. All of them share their experiences of projects they have worked and what SDLC models they select. Project complete within time estimate or selected process model effectively work or not. They share that their selected methodology for current project will applied to another project or not, it depends upon requirements of project. Methodology of combining different process models also implemented in their projects. Maximum of people are agreed that SDLC is important for project management and lack of this will create an adverse effect end-product. Some snapshots of questions are added and at last summarized results of some questions are added in APENDIX A.

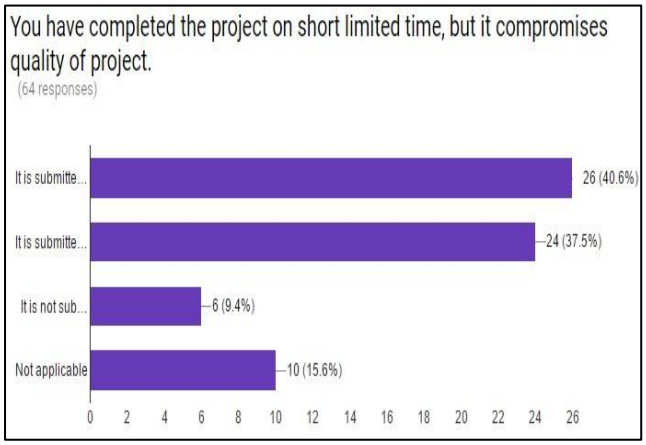

Figure 1: Responses represents Bar chart

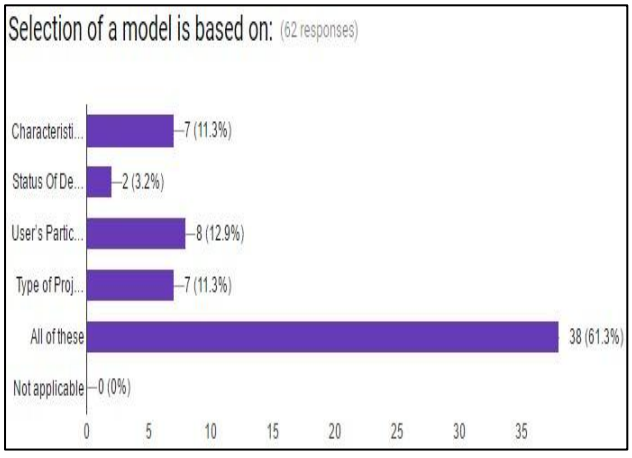

Figure 2: Responses represents Bar Chart

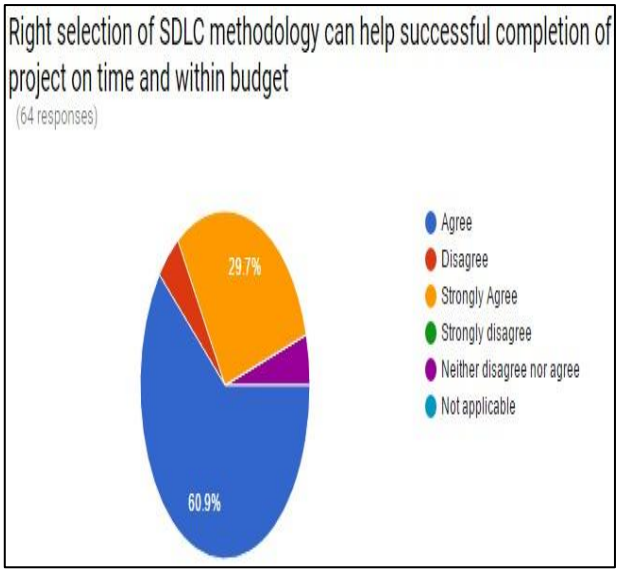

Figure 3: Responses represents Pie chart

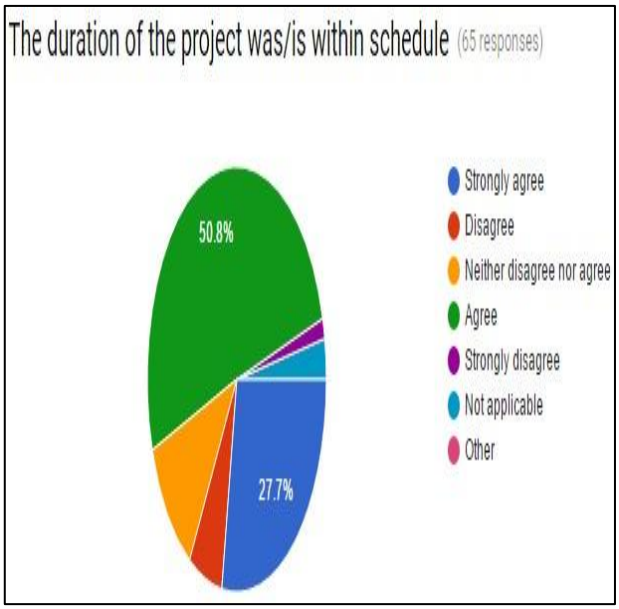

Figure 4: Responses represents Pie Chart

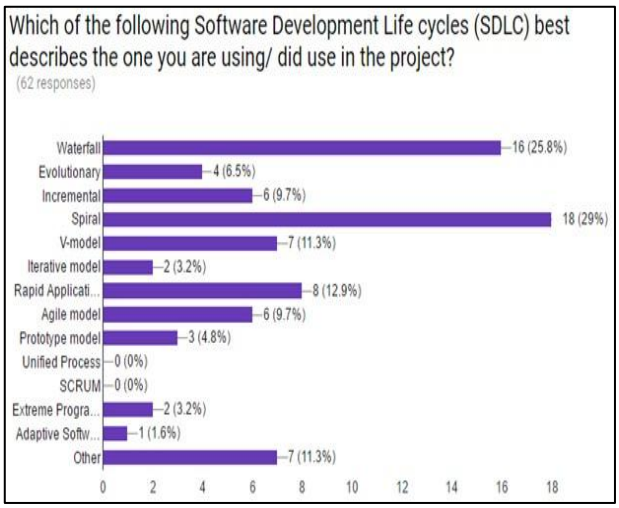

Figure 5: Responses represents Bar Chart

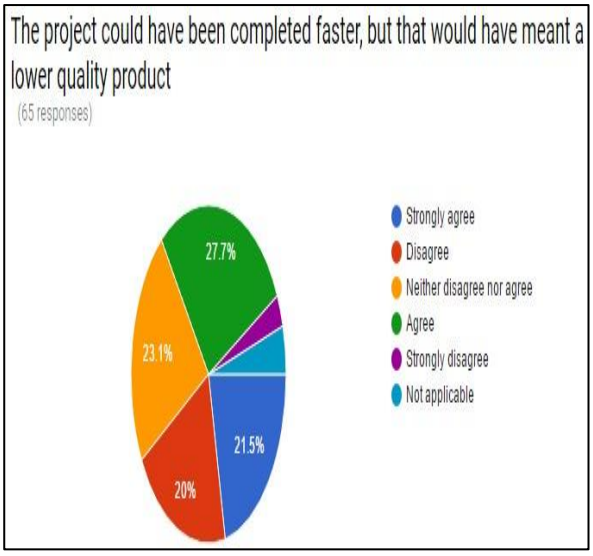

Figure 6: Responses represents Pie Chart

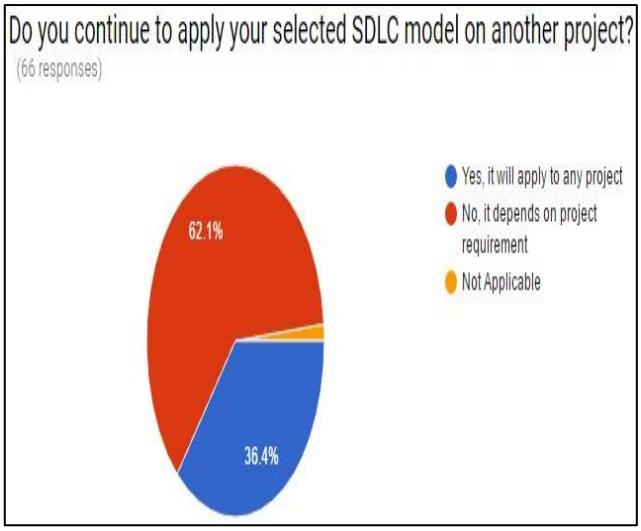

Figure 7: Responses represents Pie Chart

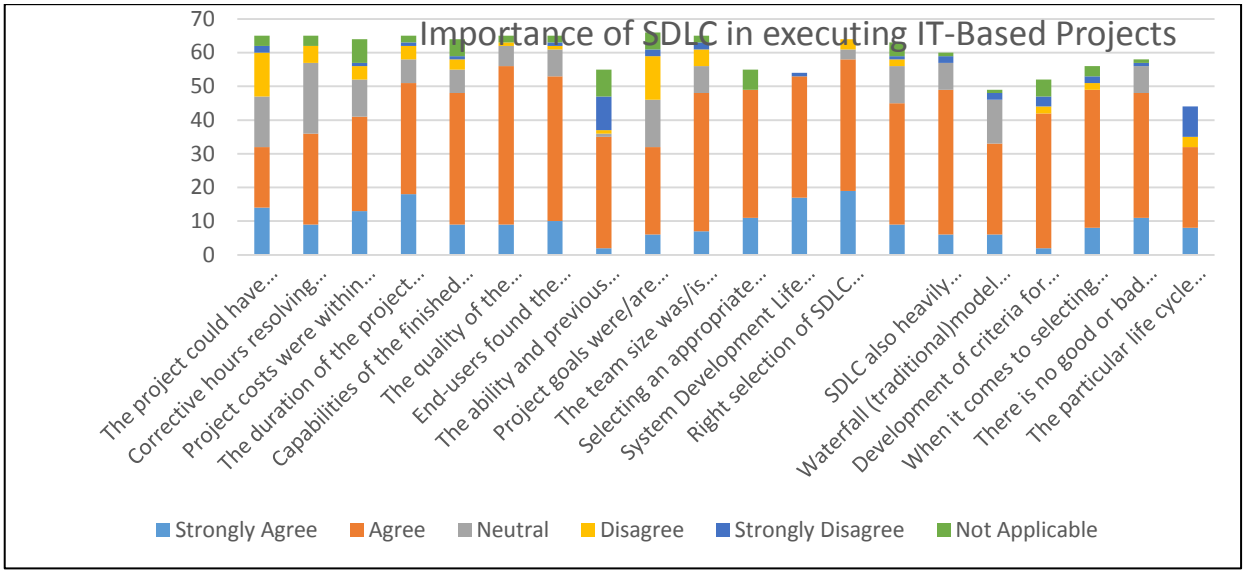

Figure 8: Summarized result represents Bar Chart

7. CONCLUSION

Based on survey results, we observe awareness of SDLC in the IT field. Maximum responses from people are in favor of using SDLC models. SDLC plays a vital role on the success of the projects. Right selection of SDLC methodology can help successful completion of project on time and within budget. Although, selecting an appropriate SDLC model is a complex and a challenging task, which requires not only broad theoretical knowledge, but also consultation with experienced expert managers. SDLC also greatly emphasizes on the use of documentation and has strict strategies on it. The most suitable methodology for your project management requirements is the best approach you should follow in project development. The specific model can significantly affect several concerns associated with a software product. If the process is weak, the end product definitely will suffer. It is necessary to promote and emphasize SDLC implementation in IT based projects. As different types of projects have different requirements, it may be required to choose the SDLC phases as per the specific needs of the project. These different requirements and needs give us various software development approaches to choose during software implementation [4]. An approach "One model fit for all" applying to SDLC methodologies is no longer appropriate. Each SDLC methodology is only effective in specific situations. Selection of model depends on the project's size, complexity, aims and objectives, the degree to which requirements are well understood, the stability of the environment in which the system will function, and most important is the customer's needs, availability during the project, and risk tolerance and the most important factor is project's timeline[6]. We identify software risk factors that covers wider range of threats through the SDLC and provide checklist that can guide project team in identifying probable risk factors and help them in designing strategies to avoid them[4]. It should not be anticipated that just because the waterfall model is the oldest original SDLC model that it is the most efficient system. In the past the model was beneficial mostly to the world of automating activities that were assigned to clerks and accountants. Nevertheless, technological evolution is demanding that systems have a greater functionality that would assist help desk operators/managers or IT specialist's experts.

8. REFERENCES

[1] S. Balaji and M. S. Murugaiyan, "Waterfall vs. V-Model vs. Agile: A comparative study on SDLC," *International Journal of Information Technology and Business* Management, vol. 2, pp. 26-30, 2012.

[2] P. Jirava and D. Series, "System development life cycle," *Scientific Papers of the University of Pardubice Series D. Pardubice: Univerzita Pardubice*, pp. 118-125, 2004.

[3] M. Tanveer, "Agile for large scale projects—A hybrid approach," in *2015 National Software Engineering Conference (NSEC)*, 2015, pp. 14-18.

[4] A. S. Shaffi and M. Al-Obaidy, "Analysis and comparative study of traditional and web information systems development methodology (WISDM) towards Web development applications," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, pp. 277-282, 2013.

[5] M. Sharma, "A Survey of project scenario impact in SDLC models selection process," *International Journal of Scientific & Engineering Research*, vol. 2, pp. 1-4, 2011.

[6] H. Hijazi, S. Alqrainy, H. Muaidi, and T. Khdour, "Risk factors in software development phases," *European Scientific Journal, ESJ*, vol. 10, 2014.

9. APENDIX A

10. Table Survey Questions

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	Not Applicable
The project could have been completed faster, but that would have meant a lower quality product	14	18	15	13	2	3
Corrective hours resolving runtime problems are/were minimal	9	27	21	5	0	3
Project costs were within budget estimates	13	28	11	4	1	7
The duration of the project was/is within schedule	18	33	7	4	1	2
Capabilities of the finished product fitted well with customer or user needs	9	39	7	3	1	5
The quality of the development team's work was acceptable	9	47	6	1	0	2
End-users found the finished product was easy to use	10	43	8	1	1	2
The ability and previous experience of the software development team was/is adequate	2	33	1	1	10	8
Project goals were/are achieved earlier than predicted	6	26	14	13	2	5
The team size was/is adequate for the project	7	41	8	5	2	2
Selecting an appropriate SDLC model is a complex and a challenging task, which requires not only broad theoretical knowledge, but also consultation with experienced expert managers.	11	38	0	0	0	6

System Development Life Cycles play a very important role on the success of the projects	17	36	0	0	1	0
Right selection of SDLC methodology can help successful completion of project on time and within budget	19	39	3	3	0	0
In SDLC, there is a possibility of combining two or more project management methodologies for the best outcome.	9	36	11	2	1	4
SDLC also heavily emphasizes on the use of documentation and has strict guidelines on it.	6	43	8	0	2	1
Waterfall (traditional)model is not capable of addressing the challenges in the modern software development domain.	6	27	13	0	2	1
Development of criteria for model selection is that one should consider employing a Decision Support Systems (DSS), expert systems(ES) or knowledge-based systems perceived as the first step towards building a system.	2	40	0	2	3	5
When it comes to selecting an appropriate Project Management Methodology, there are a few dozens of factors you should consider. Each project management methodology carries its own strengths and weaknesses.	8	41	0	2	2	3
There is no good or bad methodology and what you should follow is the most suitable one for your project management requirements.	11	37	8	0	1	1
The particular life cycle model can significantly affect various concerns associated with a software product. If the process is weak, the end product certainly will suffer.	8	24	0	3	9	0