International Journal of Computer Applications (0975 – 8887)
Volume 168 – No.8, June 2017

CloudSDLC: Cloud Software Development Life Cycle

Dhanamma Jagli
Assistant Professor
Vivekanand Education Society's Institute of Technology, Chembur, Maharashtra, India

Shireesha Yeddu
Vanasthalipuram,
Hyderabad, India

ABSTRACT

In this technology era, technology is getting change very vastly. As per the present trend in the technology cloud computing is one of the dynamic key computing and used to shared resourcefully pay for use concept in the cloud computing is much attractive and adopted by many organizational the usage of software service has been tremendously increasing that the development of different software or applications and keep available cloud user on the cloud environment. So that in this paper a new software development life cycle (SDLC) is proposed for cloud software services and is termed as CloudSDLC.

General Terms

Risk Management, Flexible, Customer Satisfaction

Keywords

SDLC, SaaS, PaaS, IaaS, SLA

1. INTRODUCTION

To build a high quality software product that satisfies the client or end user needs organizations must choose best software development life cycle (SDLC). And there are different SDLC's that can be practiced. The choice of SDLC varies from organization to organization. Perhaps, each organization has its own procedures and policies and they are also different in terms of their needs and infrastructure.

SDLC is a software development process that describes a theoretical and conceptual representation of the software development. The software development is a progression through a series of different phases such as – requirement analysis, design specification, coding or implementation, testing and maintenance. With the help of SDLC, companies can break down the work efficiently, allocate different activities to software development team, estimate budget and deadlines or time period.

To compete in global market, from last four decades, most of the companies and organization practiced and employed different life cycle models like waterfall, spiral, rapid prototype, agile and many more[1].

2. RELATED WORKS

The primary functionality of SDLC framework is to plan the activities for software development. And none of the life cycle is perfect that meets the client requirements. For many companies, selecting an appropriate and best fit SDLC for their project is a biggest challenge. This section illustrates different SDLC's and how they are used in software development.

- Water-fall Model
- Incremental Model
- Spiral Model
- V Model
- RAD Model
- Agile Model

2.1 Water-fall Model:

The waterfall model is the first and earliest among all the other life cycle models. The generic waterfall model is a cascade model, in which output of one stage comprise the input to the next stage. This model is relatively simple to use and intuitive by nature. Here, all the requirements from customer gathered at once before design, and then requirements are transformed into designs such as High level and low level designs. The designs are implemented into code in implementation phase. And implemented code is tested using test plans in testing phase. The software development process flows incrementally downwards from top to bottom like a waterfall shown in Figure 1, hence it is waterfall model [2][3][5][6].

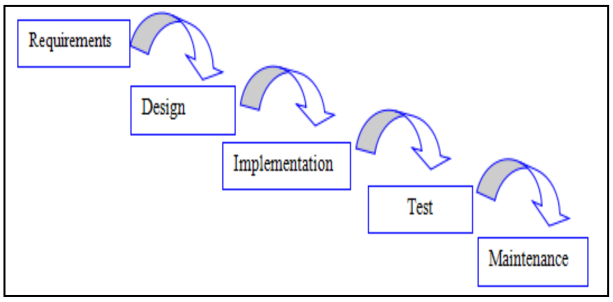

Fig 1. Waterfall Model

Advantages:

- Simple to understand and use due to cascade by nature.
- Deliverable s and milestones identified after each phase.
- Effective results with small team.

Disadvantages:

- All requirements may not be gathered or available in initial stages of project
- Delay in delivery of software project.
- Risk Management is challenging.
- Administrative cost is more.

2.2 Incremental Model

Incremental model is an intuitive approach to overcome drawbacks of waterfall model. Its development life cycle is multi-water fall model with cycle iterations, contains phases such as requirements, design, implementation and testing. Each cyclic iteration divided into small units and each unit is manageable [2][5][6]. After every iteration, a small incremental release will be produced illustrated in Figure 2. And these incremental releases of the software components and features integrated to deliver the final software product.

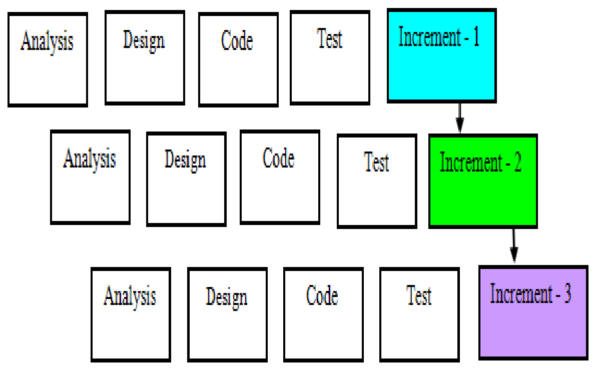

Fig 2. Incremental Model

Advantages:

- Better utilization of scarce resources in each increment cycle.
- In early stages of the development life cycle itself the business value is produced.
- Customer oriented..
- Earlier detection of problem.
- Quick and earlier release of working software module.
- Flexible enough to modify the requirements as well as scope
- Easy to test and debug in each iteration.

Disadvantages:

- As compared with other models, it more customer oriented.
- In each cyclic iteration difficult to partition functions.
- Functional and feature dependencies arises from one increment to another increment.
- Issues may arise with respect to joining iterations.

2.3 Spiral Model

The spiral model focuses more on minimizing the risk involved in project by decomposing the project into smallest units and flexible enough for modifications during the software development process. It provides more opportunities for risk assessment to evaluate risks at stage of spiral.The spiral model consists of four phases - planning, risk analysis, Engineering and Evaluation. In each iteration forms a spiral, and in each spiral, the project life cycle goes through the above mentioned four phases. In planning phase, business as well as system requirement specifications are gathered. This first spiral is known as baseline spiral. Upon the baseline spiral, rest of the spiral will be formed. Next phase, i.e, in risk analysis phase, the risk and its associated alternate solutions will be identified and implemented. At the end this phase, a prototype is produced.Third phase, i.e, Engineering phase contains development and testing both. Here, software is developed and testing performed at the end of the phase. In evaluation phase, evaluation of output is performed by user. Depending on the evaluation result project continues to next spiral [2][3][5][6].

Advantages :

- Priority is given to Risk Analysis that avoids Risk in early stages
- Suitable for large and critical projects.
- Documentation and Approvals are mandatory.
- Flexible enough to add additional features
- Software is produced early in the software life cycle.

Disadvantages:

- Costly in terms of resources
- Expertise is needed for Risk analysis
- Success of the project depends on risk analysis phase.
- Inapplicable to smaller projects.

2.4 V-Model

In software development, the V-model represents a development process that may be considered an extension of the waterfall model, and is an example of the more general V-model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes represents time or project completeness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.

V-Model (Validation and Verification Model) software development process is an extension for the waterfall model. The V-Model life cycle is not linear as compared to Waterfall model, rather the stages move upwards after the coding phase is done that forms the V shape. In V-Model life cycle shown in Figure 3, each development phase is associated with testing phase. To proceed to the next phase in the development process, the product from the previous phase validation and verification is mandatory [2][3].

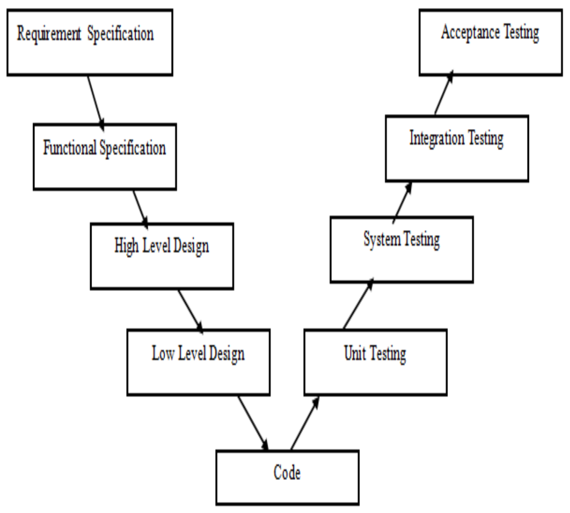

Fig 3. V-Model

Advantages:

- Success rate is more compared to waterfall model.
- Saves time due to testing of planning and design work will be finished before coding.
- Bugs are found at early stage.
- Suitable for small projects

Disadvantages:

- Least flexible with respect to modifications
- Prototype of software is not available until implementation phase
- In middle of project if any changes happens will effect changes for test documents as well as requirement documents.

2.5 RAD Model

The RAD model stands for Rapid prototyping or Rapid Application Development model. It is an incremental type model with development life cycle contains planning, designing, implementation and integration and testing. In this model, the software product and their features released in parallel as small unit increments depicted in Figure 4. The working prototype of released increments delivered and assembled in a bounded time. Quick releases of software in increments increases customer satisfaction. And customer feedback is needed to find out whether requirement specifications are met [2][4][5].

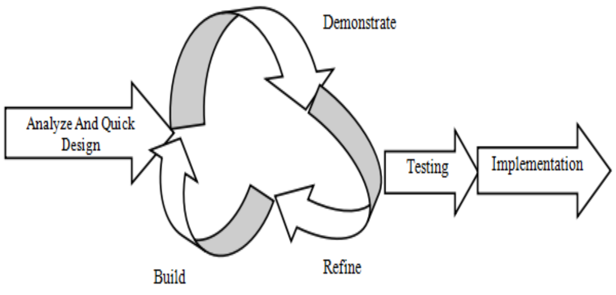

Fig 4. RAD-Model

Advantages:

- Parallel development of software components reduces time.
- Increases software components reusability
- Frequent reviews carried out
- Customer feedback needed
- Integration issues of software components addressed from initial stage itself

Disadvantages:

- Easy to build system that supports modulation and object oriented approach
- Need highly skilled software engineers or developers
- Modeling skills are mandatory
- Mostly suitable for large projects with more cost involved in modeling and auto code generation

2.6 AGILE Model

Agile Model is time-based and incremental model with rapid iterative cycles. This model provides quick software products delivery in small incremental releases by integrating lightweight processes. To develop an overall software through incremental releases the life cycle go through several iterations shown in Figure 5. And in each of the iteration there exists stages such as analysis, design, code and testing. After every iteration, the released software product is delivered to customer to take the feedback. Depending on customer feedback, if any modifications need to be done for the released product then that will be reflected in the life cycle. To ensure the quality of software product that released in each incremental stage will go through test phase. This model focuses more on customer involvement and satisfaction in software development process with high quality final product [2][7].

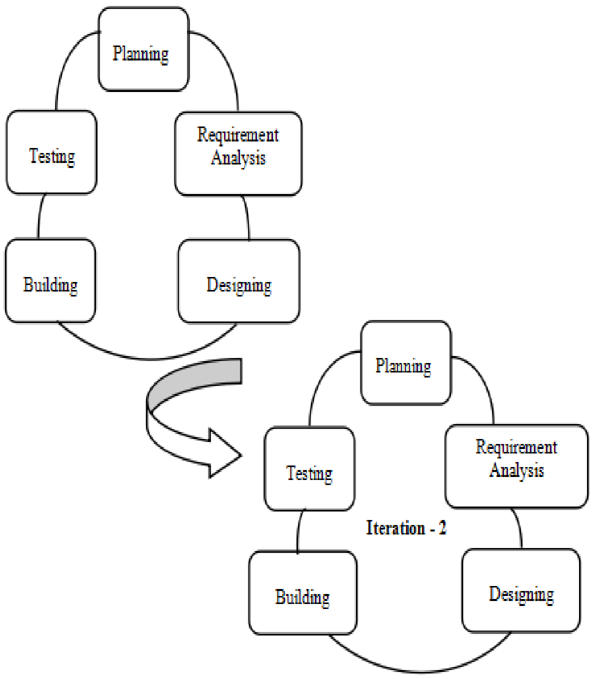

Fig 5. AGILE-Model

Advantages:

- Flexible enough to update the life cycle depending on changing requirements from user
- Customer satisfaction is more due to quick incremental releases
- Customer feedback after every release produces high quality product.
- Saves time due to brief documentation
- Less risky due to frequent customer feedback
- Quickly find and fix bugs

Disadvantages:

- Customer need to be clear about the product and its features
- Difficult for new developers in team to understand brief documentation
- Waste of time and resources and efforts too if release is not up to the customer expectation

3. SDLC MODELS COMPARISONS

The following table Table 1 shows the comparisons between different life cycle models.

Software Models	**Waterfall**	**Incremental**	**Spiral**	**V-Model**	**RAD**	**Agile**
Understanding Requirements	Understood Well At Beginning	Understood Well At Beginning	Not Understood Well At Beginning	Understood Well At Beginning	Early requirements understood well at the Beginning and need time to understand changing requirements	Early requirements understood well at the Beginning and need time to understand changing requirements
Cost	Low	Medium	High	High	High	High
Time Period	Medium	Long	Short	Medium	Short	Short
Customer Satisfaction	Low	High	High	High	High	High
Risk Involvement	High	Easy to Manage	Low	Low	Very Low	Low
Schedule Size	Within Schedule	Schedule May Exceed	Schedule May Exceed	Stick to Schedule	Stick to Schedule	Within Schedule
Flexibility	Difficult	Easy	Easy	Difficult	Easy	Easy

Table 1. SDLC Models Comparisons

4. CLOUD ENVIRONMENT

Cloud Computing is an ubiquitous computing to provide on-demand network enabled access to shared resources for users. With the help of virtualization, cloud users can even access applications and exchange data. The cloud services are classified as SaaS, PaaS and IaaS shown in Figure 6. And these services availability to users depends on Cloud deployment (i.e., public cloud or private cloud and hybrid cloud) and SLA (Service Level Agreement) between service providers and users[8].

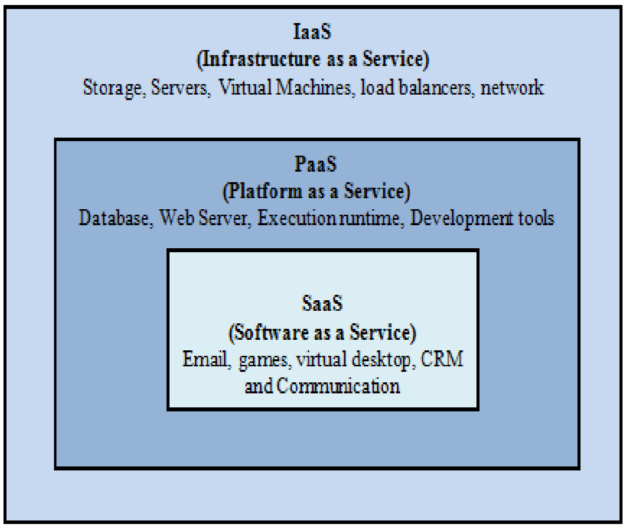
IaaS
(Infrastructure as a Service)
Storage, Servers, Virtual Machines, load balancers, network
PaaS
(Platform as a Service)
Database, Web Server, Execution runtime, Development tools
SaaS
(Software as a Service)
Email, games, virtual desktop, CRM and Communication

Fig 6. Cloud Service Models

5. PROPOSED CLOUD SaaS SDLC

Cloud Software Development Life Cycle (CSDLC) Figure 7, is different from traditional software development life cycle due to cloud models such as public, private or hybrid cloud and SLA (Service Level Agreement). SLA gives clear picture about business level policies and the roles of cloud service providers and cloud users. It provides detail description of list of services such as SaaS, PaaS, IaaS and customizability, security, accessibility and multi tenancy[8]. In CSDLC, each user can customize the application and customization information is available in SLA. Data protection and security to tackle internal and external threats is also must in CSDLC. Use of filters and firewalls prevents threats. Standard encryption and authentication techniques needed at each layer and each tenant [9]. The agreement SLA, provides access control matrix to give access to resources, application or data. Multi tenancy enables multiple users to access same instance of application on different platforms[10].

The following Figure 8 shows the proposed model for Cloud SaaS SDLC. In this proposed model, cloud service initiator initiates the request to access SaaS services after agreeing to business level policies specified in SLA. The could SaaS application services may be developed any of the conventional SDLC models such as Waterfall, Incremental, Spiral, V-model, RAD and Agile. In the below model, if any update issues related to service continuity or availability, cloud user informed through SLA.

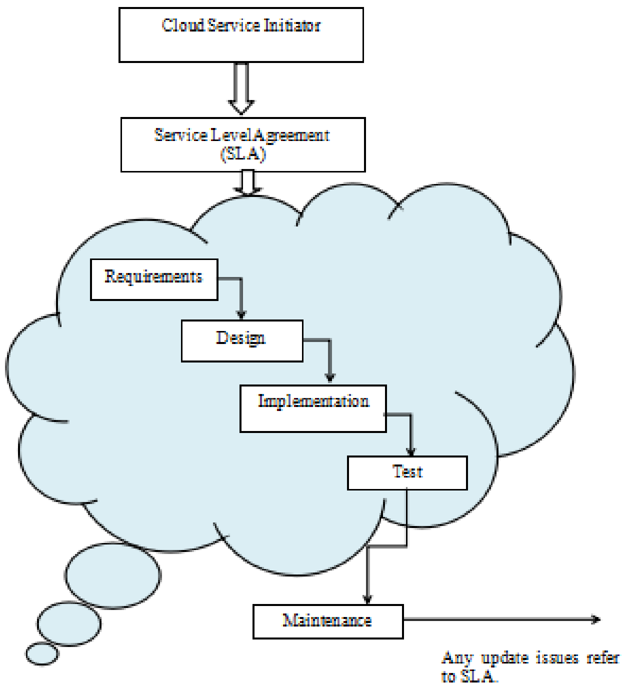

Fig 7. Proposed Cloud SaaS SDLC

6. CONCLUSION

This paper highlighted various software development (SDLC) process that describes a theoretical and conceptual representation of the software development. And also, shown the comparative approach among various software Life Cycle Models. Furthermore, specified the importance of building a high quality software product that satisfies the client or end user needs, organizations must choose best software development life cycle (SDLC). And there are different SDLC's that can be practiced. Further, depicted how to choose various SDLC models as per organization's requirement and explained each organization procedures and policies and which are different in terms of their needs and infrastructure. Through this paper described the advantages and disadvantages of most of the software process models. However, illustrated different SDLC models that help companies to break down the work efficiently, allocate different activities to software development team, estimate budget and deadlines or time period. This Paper explains the need for developing Cloud SaaS SDLC. And even explained about different SDLC models comparative matrix with different parameters. Moreover, this paper described new proposed model for Cloud Software as a Service (SaaS). Hence, this paper concludes about importance of SDLC for Cloud SaaS.

7. REFERENCES

[1] Roger S. Pressman, Software Engineering A Practitioner's Approach, 7th Edition.

[2] T Bhuvaneswari, S Prabaharan, "A Survey on Software Development Life Cycle Models", IJCSMC, Volume 2, Issue. 5, May 2013, pg.262 – 267, ISSN: 2320–088X.

[3] Ms.Shikha maheshwari, Prof.Dinesh Ch. Jain, "A Comparative Analysis of Different types of Models in Software Development Life Cycle", IJARCSSE, Volume 2, Issue 5, May 2012, pg.285-290, ISSN: 2277 128X.

[4] Mr.Srinivasan, R. Agila, “Software Development Life Cycle Model Incorporated With Clemency Brass”, IJIRAE, Volume 1 Issue 4 ,May 2014, pg.211-214, ISSN: 2349-2163.

[5] Rashmika K. Vaghela, “A Comparative Analysis of Software development life cycle Models” , IJSR, Volume 4 , Issue 6, June 2015, pg. 512-515, ISSN: 2277 - 8179.

[6] Vipul Aggarwal, Evolving a new Free-Flow Software Development Life Cycle Model Integrating Concept of Kaizen, IJARCSSE, Volume 3, Issue 9, September 2013, pg.237-243, ISSN: 2277 128X.

[7] Sheetal Sharma, Darothi Sarkar, Divya Gupta, “Agile Processes and Methodologies: A Conceptual Study”, IJCSE, Volume 4, Number 05, May 2012, pg. 892-898, ISSN : 0975-3397.

[8] S.B.Dash, H.S.Saini, T.C.Panda, A.Mishra, “Service Level Agreement Assurance in Cloud Computing : A Trust Issue”, IJCSIT, Volume 5 (3), 2014, pg. 2899-2906, ISSN - 0975-9646.

[9] Muhammad Fahad Khan, Mirza Ahsan Ullah, Aziz-ur-Rehman, “An Approach Towards Customized Multi Tenancy”, IJMECS, Volume 4, Number 9, 2012, pg. 39-44.

[10] Sunil Kumar Khatri, Himanshu Singhal, Kushbu Bahri, “Multi tenant Engineering Architecture in SaaS”, IJCA, 2013, pg.45-49, (0975-8887).