Advantages and Limitations of Different SDLC Models

Radhika D Amlani
Research Scholar
Saurashtra University,
Rajkot (Gujarat)

Abstract

Software engineering is the area which is constantly growing. It is very interesting subject to learn as all the software development industry based on this specified area. Now a days, there are lots of software development life cycle models available. According to the requirements, software industry people are using it as per their requirements.

As there are lots of SDLC models, they are used according to their requirements. So, it is needed to know the requirements in which the SDLC models is used.

This paper is about the pros and cons of some models. So, user can take the advantage of this paper to find the model best suitable for their need.

Keywords

SDLC Phase, Advantages of the SDLC models, Limitations of the SDLC models, Pros and cons of different SDLC models.

INTRODUCTION

Software Development Life Cycle

Software Development Life Cycle Model is used as a process of creating and altering current existing system. SDLC used in information system, systems engineering, and software engineering. SDLC can be thought of as a concept that used by many software development methodologies, which are currently available in market or software industry. SDLC provide a framework to create, plan and control any information system to be developed.

There are so many SDLC based, software engineering models available in market now-a-days. Depending upon the suitability, the software engineering model can be used to put forward any software project. Each of the methodologies or models has different level of risk and benefits to cope with the project requirements, budget and estimated completion timeline. There are models which are suitable for large project, where some focus on lightweight process that allow rapid changes throughout whole software development life cycle.

The pros and cons of the different SDLC models are given below.

Pros and Cons of SDLC Model

Advantages of Waterfall Model:

1) Require business needs and requirements in beginning.
As the analysis team determines the business needs and requirements first, this process facilitates to better cope with the organizations need.
2) This process defines definite starting and ending points of a project.
3) Early detection of errors
As all the stages are clearly defined so, this process ensures early detection of errors and misunderstanding in its each stage.
4) Require Documentation
Requirement specification document serve as the guideline for the development and testing phase.
5) In future, for code revision and future project enhancement these documents are useful in this process.
6) Since the following phases are dependent on previous phase, this approach ensures project deadline control.
7) Each phase is discrete and team members involved in a stage ensures the perfection of the stage before delivering to next stage. Waterfall process ensures greater project output.
8) This approach can be very efficient when team members are dispersed in different locations.
9) The amount of resources required to implement waterfall model is lower than other methods.

Limitations of Waterfall Model:

1) The greater disadvantage of this approach is that there is no way to go back. Once a stage is complete means it is locked.
2) Sometimes it's really tough to estimate time required for different phases and incorrect assumption can fail the project to meet its deadline.
3) Waterfall model does not allow changes as per client's requirement, so it is less flexible.
4) If changes are to be made in waterfall process, the project has to be started all over again. This can be expensive for some organization.
5) Increase software development time and expense since client keep adding requirement on the list.
6) Team members of rest of the phase sits idle except the team member who are under the current working phase.

Advantages of Spiral Model:

1. Spiral life cycle model is one of the most flexible SDLC models. Development phases can be determined by the project manager, according to the complexity of the project.
2. Project monitoring is very easy and effective as it is done in each phase of each iteration. This monitoring is done by experts or by concerned people. This makes the model more transparent.
3. Risk management is key feature of this model, which makes it more attractive compared to other models.
4. If changes are introduced at later stage in life cycle, coping with these changes isn't a very big problem for the project manager.
5. It is suitable for high risk projects, where business needs may be unstable.
6. A highly customized product can be developed using this model.
7. Since the prototype building is done in small fragments or bits, cost estimation becomes easy and

the customer can gain control on administration of the new system.

8. As the model continues towards final phase, the customer's expertise on new system grows, enabling smooth development of the product meeting client's needs.

Limitations of the Spiral Model:

1. Cost involved in this model is usually very high.
2. It is a complicated approach especially for the project with a clear SRS (System Requirement Specification).
3. Rules and protocols should be followed properly to effectively implement this model. Through-out the project life cycle development, it is very hard to follow rules and protocols.
4. It is not suitable for low risk projects.
5. Meeting budgetary and scheduling requirements is very tough with this software development process.
6. Due to various customizations allowed from the client, it is very hard to use same prototype in other projects.
7. It needs extensive skill in evaluating uncertainties or risks associated with the project and their abatement.
8. The models work best for large projects only, where the costs involved are much higher and system pre requisites involves higher level of complexity.
9. Risk assessment expertises are required.
10. Spiral may continue indefinitely.

Advantages of the V-Shape Model:

1. Simple and easy to use
2. Testing activities are planned before coding. This saves a lot of time and also helps in developing a very good understanding of the project at the beginning state.
3. Each phase has specific deliverables.
4. Works well for where requirements are easily understood.
5. Works for small projects.
6. Higher chance of success because of the development of test plans early on during the life cycle.
7. The model encourages verification and validation of all internal and external deliverables. Not just the software products.
8. The V-shaped model encourages definition of the requirements before designing the system and it encourages designing the software before building the components.
9. It defines the products that the development process should generate, each deliverable must be testable.

Limitations of the V-Shape Model:

1. It is inflexible; it has no ability to respond to change. It is very rigid.
2. It produces inefficient testing methodologies.
3. If any changes happen in mid way, then the best documents along with requirement document has to be updated.
4. Adjusting scope is difficult and expensive.
5. Software is developed during the implementations phase, so no early prototypes of the software are produced.
6. Model doesn't provide a clear path for problems found during testing phases.
7. It doesn't handle concurrent event.

Advantages of the Rational Unified Process

1. RUP is a complete methodology to manufacture software.
2. It is process with complement document facility.
3. RUP is openly published, distributed and supported.
4. It supports changing requirements to meet its desired software.
5. It supports iteration process so we can integrate the code in development life cycle in lesser time and effort spent in integration.
6. The reuse of code easy and faster so development time is less.
7. There is online training and tutorial available for this process.
8. Debugging is very easy due to component base architecture.

Limitations of the Rational Unified Process

1. The team members need to be expert in their field to develop the software under this methodology.
2. The development process is too complex and disorganized.
3. On cutting edge projects which utilise new technology, the reuse of components will not be possible. Hence the time saving one could have made will be impossible to fulfil.
4. Integration throughout the process of software development, in theory sounds a good thing. But on particularly big projects with multiple development streams it will only add to the confusion and cause more issues during the stages of testing.
5. It's too complex to implement, and too difficult to learn.
6. May lead to undisciplined form of software development.

Advantages of the Prototype Model:

1. When prototype Model is shown to the user, he gets a proper clarity about his requirements. And feel the functionality of the software, so can suggest the changes and modifications.
2. This type of approach of developing the software is used for non-IT literate people. They usually cannot explain their requirements specifically.
3. When client is not confident about the developer's capabilities, he asks for a small prototype to be built. Based on this prototype model, he can judge capabilities of developer.

4. It helps to demonstrate the concept to the investors to get funding for project.
5. It reduces risk of failure, as potential risks can be identified early and steps can be taken to remove that risk.
6. Constant interaction between development team and client provides a very good and co-operative environment during project.
7. Time required to complete the project after getting final SRS is reduces as the developers has a better idea about how he should approach the project.
8. The customer does not need to wait long for working software.
9. Feedbacks from customer are received periodically and the changes don't come as a last minute surprise.

Limitations of the Prototype Model:

1. Sometimes the start-up cost of building the development team, focused on making prototype is high. And usually in starting Prototype is done at the cost of the developer. So it is done using minimal resources.
2. It is a slow process.
3. Once we get proper requirements from client after showing prototype model, it may be of no use.
4. Too much involvement of client is not always preferred by the developer.
5. Too many changes can disturb the development team.
6. Customer could believe the prototype as the working version.
7. Developer also could make the implementation compromises where he could make the quick fixes to the prototype and make is as a working version.
8. Often clients expect that a few minor changes to the prototype will more than suffice their needs. They fail to realise that no consideration was given to the overall quality of the software in the rush to develop the prototype.

Advantages of the Iterative and Incremental Model:

1. The versions are provided after iteration of the incremental model.
2. After using first iterated model, user can give their suggestion and demand for changes.
3. This model does not affect anyone's business values because they provides core of the software which customer needs after first iteration. To the customer software will help him to keep running his business.
4. It is flexible to the customer's requirements and easy to manage model.
5. Better risk management is there in this model because one can confirm the outcome by the customer after every version because every version is prepared according to the plan.
6. Easy to test as testing is done in iteration as per requirement.
7. This model is used when requirements are clear to some extend but project scope requires pure linear approach.
8. Complete implementation by decided dead line.
9. Sometimes early increments can be implemented with fewer people.
10. Lover risk of project failure compared to other approaches.
11. Results are obtained early and periodically.
12. Parallel development can be planned.
13. Progress can be measured by setting milestone.
14. Testing and debugging during smaller iteration is easy.

Limitations of the Iterative and Incremental Model:

1. Each phase of an iteration is very rigid and do not overlap each other.
2. Problems may arise related to system architecture because not all requirements are gather in initial stage of the development process.
3. Each increment needs to be relatively small.
4. Mapping requirements to increments may not be easy so managing documents are very difficult.
5. Common features of the software are difficult to identify.
6. During development process changes are being done at first iteration. As if it continues to change and it never finished.
7. More resources may be required.
8. More management attention is required due to frequent changes in requirements.
9. Does not allow iterations within an increment.

Advantages of the Rapid Application Development:

1. Working software is available much earlier than any conventional method.
2. RAD produces systems more quickly and to a business focus, this approach tends to produce systems at a lower cost.
3. A greater level of commitment is there from stakeholder. So user are seen as gaining more of sense of ownership of a system, while developers are seen a gaining more satisfaction from producing successful systems quickly.
4. Focus on essential system elements from user viewpoint.
5. Provides the ability to rapidly change system design as demanded by users.
6. Gives tighter fit between user requirements and system specifications.
7. Saving time, money and human efforts.
8. Changing or stopping the course of development on a product that is not meeting its objectives.
9. Resulting final product often match user's needs and expectations very closely.

Limitations of the Rapid Application Development:

1. More speed and lower cost may lead to power overall system quality.
2. Due to inappropriate information, developed system might be misaligning.
3. Project may end up with more resources than needed.
4. Due to hurry, there may inconsistent designs within and across system.

5. There may be violation of programming standards related to inconsistent naming conventions and inconsistent documentations.
6. There can be lack of scalability in design.
7. Difficulty with module reuse for future systems.
8. High cost of commitment on the part of main user personnel.
9. Formal reviews and audits are more difficult to implement than for a complete system.
10. Tendency for difficult problems to be pushed to the future to demonstrate early success to management.
11. RAD prototyping can be difficult to manage in large organizations.
12. User can be misleading to adopt premature working prototype as the finished product.

Advantages of the Joint Application Development:

1. Improved the communication between business users and the project team.
2. Enhances quality.
3. It allows for the simultaneous gathering and consolidating of large amounts of information.
4. The experts get a chance to share their views, understand views of others, and develop the sense of project ownership.
5. Creates a design from the customer's perspective.
6. It reduces organizational infighting. By bringing all the decision makers together to design the system, JAD brings conflicting objectives and hidden agendas to light early in the project, when they can still be addressed effectively and before they've had time to do much damage.
7. Project teams get focused and stay focused.

   In the workshop, the participants will build a common view of the project and a common language to discuss the issues. These elements will stay with the team for the life of the project.
8. A natural partnership with modern development tools.

   JAD helps realize the full potential of today's powerful development tools by providing high-quality input requirements quickly.
9. Improves design quality.

   The JAD improves the quality of the deliverable of the design phase because it forces a definition of that deliverable in advance. During the workshop the participants are all focused on a common goal. Users in the workshop will have a better understanding of the business issues, the systems issues, and the volume of work to be done.
10. Enhanced education for participants and observers.

    By participating in JAD and be the medium between other users and IT, the business end-users will be kept fully informed about the progress of the system development.
11. Enables rapid development.

    In JAD, information can be obtained and validated in a shorter time frame by involving all participants who have a stake in the outcome of the session.
12. Improved quality of deliverables—quality assurance of deliverable is 'built-in' to a large extent.

    Much of the system's quality depends on the requirements gathered. JAD involves users in the development life cycle, lets users define their requirements, and thus ensures that the system developed satisfies the actual activities of the business.
13. Reduced system cost.

    Much of the system development cost is in terms of man-hours of both system developers and business users involved. Reduced development time reduces the labour cost for developers, as well as users. JAD can reduce the involvement time of business experts and hence reduce the cost further. Cost is also reduced by catching errors, misunderstandings and mistakes early in the development phrase.

Limitations of the Joint Application Development

1. JAD is more expensive and can be cumbersome if Project is large.

   If the group is too large relative to the size of the project then it would be very difficult to manage that group in JAD workshop.
2. Highly expert people are required to manage JAD Project.

Advantages of the Scrum:

1) Agile scrum helps the company in saving time and money.
2) Scrum methodology enables projects where the business requirements documentation is hard to measure to be successfully developed.
3) Fast moving developments can be quickly coded and tested using this method, as a mistake can be easily rectified.
4) It is a lightly controlled method which insists on frequent updating of the progress in work through regular meetings. Thus there is clear visibility of the project development.
5) Like any other agile methodology, this is also iterative in nature. It requires continuous feedback from the user.
6) Due to short sprints and constant feedback, it becomes easier to cope with the changes.
7) Daily meetings make it possible to measure individual productivity. This leads to the improvement in the productivity of each of the team members.
8) Issues are identified well in advance through the daily meetings and hence can be resolved in speedily.
9) It is easier to deliver a quality product in a scheduled time.
10) Scrum can work with any technology/ programming language but is particularly useful for fast moving web technology or new media projects.
11) The overhead cost in terms of process and management is minimal thus leading to a quicker result.

Limitations of the Scrum:

1) Agile Scrum is one of the leading software development model because if there is not a definite end date, the project management stakeholders will be used to keep demanding new functionality to be delivered.
2) If a task is not well defined, estimating project costs and time will not be accurate. In such a case, the task can be spread over several sprints.
3) If the team members are not committed, the project will either never complete or fail.
4) It is good for small, fast moving projects as it works well only with small team.
5) This methodology needs experienced team members only. If the team consists of people who are new to this technology, the project cannot be completed in time.
6) Scrum works well when the Scrum Master trusts the team they are managing. If they practice too strict control over the team members, it can be extremely frustrating for them, leading to demoralisation and the failure of the project.
7) If any of the team members leave during a development it can have a huge inverse effect on the project development
8) Project quality management is hard to implement and measure if the test team are not able to conduct failure testing after each sprint.

Advantages of the Extreme Programming

1) Constant feedback from customer

Developer gets constant feedback from customer which helps them to add new features to the system being developed.

2) Customers are available onsite

Customers are constantly available on site. So, when developer team required asking anything it would be very easy.

3) Pair programming gives better coding

As programming is done in a team of two to three people, it would be beneficial to the coding. Because all the people in a team have different way of thinking relates to see a particular programming aspect. So better coding would be outcome of these aspects.

4) Refactoring is also beneficial

Changes to the coding are admirable in XP. Instead of designing the entire system up front, design as you go, making improvements as needed. So the interface would not change.

5) Continuous Integration

As new code is developed, it is tested and integrated with the old code. The entire code base is constantly being rebuilt, and retested in an automated fashion.

6) Silly mistakes are quickly caught

Simple mistakes such as syntax errors or repeated variable names can be easily caught and fixed right then and there. This might not seem like much, but it can cut down debug time later and prevent small irritating bugs.

7) Better concentration

This isn't particularly scientific, but it seems that too people working have a lesser tendency to get distracted. This results in shorter development times too.

8) Its a good training ground for large software projects

Very little software is written by one person nowadays, teams ranging from small to big are not just the standard, they are a necessity. Pair programming teaches you a lot of the soft skills you'll need: tolerance, respect, understanding and responsibility.

9) Combining knowledge

Computer science is a vast field and my course is rather fast-paced. It's hard for any single student to have a complete knowledge of what's been going on in class, so two of them working together means that they can pool their knowledge and work together.

10) Constant planning

The problem with most projects is they attempt to plan once and assume they are done with it. They spend a lot of time up front to come up with a date that everyone knows is bogus. XP takes the view that planning is good, but you really want to be planning and re-planning all the time. Reality always interrupts the plans.

11) It gives customers the ability to see whether or not a company can deliver on its promises.

12) It gives management many tools, including predictability, flexibility of resources, consistency, and visibility into what's really going on.

Limitations of Extreme Programming:

1) Skill disparity

This is the number one potential problem. If the partners are of completely different skill levels, you might have one programmer doing all the work or constantly tutoring the other. This is ok if one wants to set up a teacher-student relationship or are introducing a new programmer to your system, but if not, it can defeat the entire purpose of XP.

2) Not actually getting the work done

For some people pair programming sessions can easily generate in socializing sessions. There are some people who don't work when there is someone next to them examining their work, these people will not benefit much either.

3) Developer egos

This is something that is not likely to happen in a classroom setting, but in more experienced teams, each programmer might try to push their own ideas of how things should be done, both of which may be perfectly valid. These sorts of conflicts can be totally unsuccessful.

4) Could not release in time

As members are working in teams, they might not be able to reach the target in time or complete the iteration in defined timebox. It might lead to late delivery of final product.

5) Required very good management

If management is not done properly then it would be very tough to get work done form the all team members.

Conclusion:

In this paper, Researcher has tried to give pros and cons of the different SDLC models. As all the models have their usability and limitations, so where it can be applicable is decided by its usability. This paper is useful for the developer to choose the SDLC model according to their requirements. In this paper researcher has taken 10 different SDLC models.

REFERENCE:

[1] www.en.wikipedia.org/wiki/Systems_development_life-cycle

[2]www.en.wikipedia.org/wiki/SDLC

[3]www.waterfall-model.com/sdlc/

[4]www.condor.depaul.edu/jpetlick/extra/394/Session2.ppt

[5]www.searchsoftwarequality.techtarget.com/.../systems-development

[6] Sharma, Manik, and Gurdev Singh. "Static and Dynamic metrics-A Comparative Analysis." Emerging Trends in Computing and Information Technology (2011).

[7]http://searchsoftwarequality.techtarget.com/answer/Software-development-life-cycle-phases-iterations-explained-step-by-step

[8]www.allinterview.com › Categories › Software

[9] www.vtlglobal.com/software-development-life-cycle.html

[10]www.e-digg.com/services/software/**sdlc**-life-cycle.html - United States

[12] Sharma, Manik, et al. "Comparative study of static metrics of procedural and object oriented programming languages." International Journal of Computers & Technology 2.1 (2012): 15-19.

[12]www.shazsoftware.com/software-development-life-cycle.html