­

2019

Insights on system lifecycles, comparisons & contrasts between agile and traditional approaches, usable approaches and best choice explanation.

**IDM Negombo - K.P.I. Shenesh Perera**

System Analysis & Design

Report #1

RKTC-IDM

Contents

[System Development Life Cycle 4](#_Toc11686713)

[What is SDLC? 4](#_Toc11686714)

[Feasibility Study 5](#_Toc11686715)

[Analysis 5](#_Toc11686716)

[Design 5](#_Toc11686717)

[Implementation 6](#_Toc11686718)

[Maintenance 6](#_Toc11686719)

[Traditional & Agile approaches 7](#_Toc11686720)

[What is traditional project management? 7](#_Toc11686721)

[What is agile project management? 8](#_Toc11686722)

[Traditional or agile? 9](#_Toc11686723)

[Lifecycle modals that can be used at IFRB 10](#_Toc11686724)

[Waterfall 10](#_Toc11686725)

[Six Sigma 11](#_Toc11686726)

[Scrum 11](#_Toc11686727)

[Kanban 12](#_Toc11686728)

[What is the best approach IFRB can follow? 13](#_Toc11686729)

# System Development Life Cycle

## What is SDLC?



Figure 1.0, sdtimes.com, Sandra Lupanava, 16/10/2017

The software development life cycle is a set of procedures that developers go through during the development of a new system. The system development life cycle is important because this organizes the amount of money and effort that goes into a particular system. It might not seem crucial for an individual but as a whole, where a company is involved, where everything is measured by how much profit comes your way, it is very crucial.

Following the SDLC, makes sure that you do not put too much or too less effort into a particular system and also makes sure that you will not back-track your way to start everything from scratch because you’ve come to a point in your development where your existing solution doesn’t match the requirements completely.

There are 5 main steps in the SDLC:

1. Feasibility Study
2. Analysis
3. Design
4. Implementation
5. Maintenance

Ofcourse, this is just a subject matter, there may be more steps than this in an actual SDLC, but never less than this. Every **good** developer goes through these main 5 steps even without intention, and leads to good clean code and a product that is delivered in time.

## Feasibility Study

This is the first step of every SDLC and is often the step that most developers and development companies skip intentionally or unintentionally which results in a bad product that does not meet the client’s requirements, not because enough money or time went to it, it is just because the team that worked on it did not have the potential to grasp a project of that complexity.

This step is an investigation step, where analysts look at an existing system to identify all core problems and then consider if it is tactically possible to solve and strategically cost effective to solve through the introduction of a possibly better computer based solution.

A good feasibility study must contain a description of the existing system (the whats and hows), set of statements describing all problems with the existing system, the objectives of the new system, possible alternative solutions to existing problems, the factors that have been considered and possibly a strategically and tactically reasonable course of action.

## Analysis

This is the stage that every company attempts their best at but fail to go deep enough. This is also another investigation step just like the feasibility study, however in this step you focus more on the existing system’s problems.

System analysts will attempt to use fact-finding techniques like:

* Questionnaires
* Interviews
* Observation
* Examining log files
* Consulting management

Then through these fact-finding techniques, in the analysis stage, analysts very clearly identify all existing problems then document them accordingly.

## Design

This stage is a much more complex stage than the previous stages and the stage that I believe is the most important in any SDLC. The outcome of this stage is what the developers and the rest of the staff will utilize to develop a system. As such any caveats, tradeoffs, pros/cons and problems in this design will directly be reflected in the final product.

I believe that IFRB must put their best foot forward and delve as deep as possible to grasp the best, cleanest, flexible and the most scalable design you can come up with. This will help ensure that the final product is not only perfect but also something that can be produced within the given period for the offered money.

In the design stage you’re primarily concerned with things like:

1. Input/Output
2. Data Storage
3. UI/UX
4. Backup and fail-safe recovery procedures
5. Security
6. Licensing
7. Standards
8. Test plans

## Implementation

This stage becomes extremely simple to execute given the fact that the previous steps were executed properly. If for some reason, the previous stages weren’t done properly, this stage is where you will lose the most time and most money resulting in a bad solution and unhappy clients which I believe is what IFRB is facing right now.

In this stage, you give the flesh to the skeleton that you put together during the design stage. This means that you will:

1. Be involved in development
2. Do code review
3. Have strategy meetings
4. Test your codebase for errors
5. Ensure that your implementation **does not in any way or form go beyond the design**.
6. Create User and System Documentation.
7. Deploy your solution to targeted environment to test for compatibility.
8. Conduct post-implementation reviews.

## Maintenance

This is the final stage in SDLC, the point where you polish and clean up any loose ends in the implementation. By this stage you must have a working solution that covers atleast 75% of all user and system requirements, run with no application-breaking bugs/errors and is perfectly presentable to your client.

Often in this stage, most companies attempt to change implementation details or try to rewrite them. **Do not do this.** This may result in breaking your existing solution.

After this stage, per your clients’ needs you are entirely capable of restarting the whole cycle from the beginning.

# Traditional & Agile approaches



Figure 1.1, zestard.com, zestard blog, 27/8/2015

## What is traditional project management?

Traditional project management (TPM) is a standard procedure for projects to be handled in a sequential manner (one after the other). The main stages you will see in many such approaches will be:

1. Initiation
2. Planning
3. Execution
4. Monitoring
5. Completion

Traditional project management emphasizes its processes to be linear, documented, planned and appropriately prioritized. The TPM method revolves around the concept that time and budget are variables but requirements are fixed. As a result of this concept, advocates of the TPM method face time and budget issues. For each step, there is a standard that is defined officially by the PMBOK company, which are followed word by word by every project manager.

Despite the flaws in this approach, there are some notable benefits one can observe in TPM:

1. Extremely clear objectives
2. Highly controllable procedures
3. Concise documentation
4. Higher accountability within staff

## What is agile project management?

This approach is now the most famous and the most used method all around the world. All companies have realized the benefits in Agile development, as such it has become a priority for all DevOps staff to prioritize in their path of learning.

So what makes agile project management so revolutionizing? It’s teamwork. Agile project management at its core relies heavily in the combination of efforts within a team. Collaboration and time boxing difficult tasks and the sheer flexibility in responding to changes as quick as possible is what has made agile project management to triumph over traditional project management.

The 4 main golden rules in agile project management are:

1. Higher focus on each individual than the processes or the tools involved
2. Development is more important that documentation
3. The collaboration with clients is more important than negotiating with them
4. The process **must** be able to quickly respond to change than following an absolute plan

Now with these 4 rules it must become obvious what benefits there are in following agile project management:

1. Higher flexibility in terms of prioritization than traditional approach
2. Much more predictable and earlier delivery
3. The budget, project scheduling and the timeline is more predictable
4. Quality is much more improved
5. Higher transparency due to high flexibility
6. Employees feel more involved leading to productive hours of work

Every agile approach has one thing in common. The tasks are iterative. Each task in a project is divided into short “sprints” which take less time. Unlike TPM, planning and prioritization aren’t crucial so agile is more flexible in terms of changes.

# Traditional or agile?

Go through the following table, each approach is better in certain characteristics and aren’t as good in some.

|  |  |  |
| --- | --- | --- |
| **Characteristics** | **Agile** | **Traditional** |
| Organizational Structure | Iterative | Linear |
| Scale | Small to medium | Large |
| Development model | Evolutionary | Cycle |
| Preference | Adaptation | Anticipation |
| Focus | Product | Process |
| Client involvement | High | Low |
| Testing | Flexible | Sequential |

As you see in the above table, in most situations agile is more favorable than traditional, as such it is more widely used. Although it is said in the table that agile can only be used for small to medium scale teams, there are specialized agile approaches that are designed for large scale teams, these approaches aren’t as popular as traditional is to large teams.

# Lifecycle modals that can be used at IFRB

## Waterfall

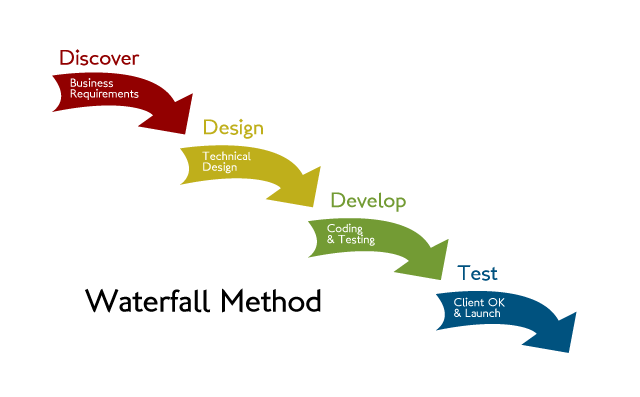


Figure 1.2, waterfall-agilemethodology.blogspot.com, Ravi Agrawal, 06/09/2013

This is a traditional project management approach that gives focus to being linear and sequential over anything else. This is the earliest SDLC approach known to software development and have quite a lot of solutions to existing caveats despite the fact that it’s traditional approach.

The waterfall method has the following requirements:

1. Clear documentation
2. Product defining must be stable
3. Technology used cannot be dynamic
4. There are no vague requirements
5. Project must be short

Some advantages and disadvantages are:

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Simple and very easy to comprehend | The solution is not bought to a working point till the end |
| Properly stated milestones | Higher risk and uncertainity |
| Clearly defined stages | Bad for OOP based development |
| Processes and the results are very well documented | Cannot respond to change. |

## Six Sigma

This is an evolution of a traditional project management modal that was first put forward in 1986 by the engineers at Motorola. The primary focus in this modal is to reduce the quantity of errors in a process by what’s not working and then removing it. As a result, it is now considered an obsolete and isn’t heavily used anymore.

The key requirements in six sigma are:

1. Defining the problems and the project’s goals
2. Measuring various aspects in the current process
3. Analyze data then find the biggest defects in the process
4. Control how the process is repeated in the future

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Ideal for small scale projects | More time is spent in development |
| Debugging is as simple as identifying, removal and re-implementing | Unclear as to what’s done |
| Documentation is concise | Repetition might take place |

## Scrum

One of the most famous agile approaches out there in the industry. It is so famous due to it’s prioritization on development, delivery and sustaining extremely complex products through collaborating iteratively with the whole team accounted for it.

The key requirements of Scrum are:

1. Extreme commitment
2. Openness within team members
3. Respect among team members
4. Focusing on difficult tasks as a team
5. Courage to take on difficult tasks as a team.

There are what’s called “events” in scrum approaches:

* Sprint: Iteratively time boxing to achieve a goal. The time frame of each sprint can not exceed a month and must be consistent throughout the development process.
* Sprint planning: The whole team must get together to plan the next sprint
* Daily Scrum: Usually a 10 minutes’ time boxed meeting that is held at the same time every day of a sprint to discuss achievements and to layout expectations.
* Sprint review: A casual meeting held at the end of every sprint where the teams present their achievements, feedback and discuss with stakeholders.
* Sprint retrospective: A formal meeting where the team reflects on the finished sprint and then decide how they can improve in their next.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Team-focused, everybody enjoys working | Lack of priority to documentation |
| Very efficient in terms of delegating tasks | Less focus on process control and planning |
| Extremely cost-effective | Recommended for teams that are usually small |

## Kanban

Kanban is also another very popular agile approach, similar to scrum in many ways. It was introduced by Toyota in the 1940s. It’s focus is on being extremely visual with regard to every process that’s involved in the development.

The key requirements for Kanban are:

1. Concise visualization
2. Putting limits to work in progress
3. Managing the general flow
4. Feedback loops
5. Collaborative evolution

As I mentioned, Kanban is extremely visual so generally a drawing of what’s known as a Kanban board is used, in which Kanban cards isolate the tasks and Kanban swim lanes for better organization.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Continuous Integration | Lack of focus on documentation |
| Extremely visual | Specific tools are required to draw visually appealing Kanban boards |
| General sense of comprehension within the team |

# What is the best approach IFRB can follow?

I personally say that Scrum is the best approach that IFRB is most suitable for. **I believe that the main reason why the IFRB teams can’t produce quality products on time is because of communication barriers or the lack of communication.** As such, there must be a higher focus on individual team members and the team workflow as a whole than the process, tooling and what not. Because clearly, IFRB is capable of producing quality content, just not within the time frame.

How will Scrum break the communication barrier? One of the biggest and the most alluring things about scrum is the fact that it gives priority to holding lots of meetings. Through these meetings, the individual team members will identify their problems, notify each other of delays, set expectations, clearly identify what needs to be done and what doesn’t need to be done, give priority to having a general sense of what your role is within the team and overall lead to efficient team that will put communication before all else.

With all these factors combined and communication playing a key role within the team, I suppose that the teams will be more flexible with their clients and cooperate with clients than negotiate with them, thus eliminating needs for making unnecessary changes or wasting time on altering bits and pieces.

This way I believe, IFRB teams can move forward and fast delivering quality products in the expected time frame.

|  |  |
| --- | --- |
| **Bibliography** | |
| Reddit DevOps | <https://www.reddit.com/r/devops/> |
| Home of Scrum | <https://www.scrum.org/> |
| Atlassian Kanban | <https://www.atlassian.com/agile/kanban> |