
# Django Rest Framework

## Notes By

# Naveen

# SDLC

**What is SDLC ?**

SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time.

SDLC includes a detailed plan for how to develop, alter, maintain, and replace a software system.

SDLC involves several distinct stages**, including planning, design, building, testing, and deployment**.

Popular SDLC models include the **waterfall model, spiral model**, and **Agile model**.

## SDLC Phases

- Requirement gathering and analysis
- Design
- Implementation or coding
- Testing
- Deployment
- Maintenance

## Requirement analysis:

In this phase all the requirement are collected from customer/client. They are provided in a document called Businessmen requirement specification (BRS) and System requirement specification (SRS). All the detail are discuss with customer/client in detail.

## Design:

It has two steps:

- **High level design (HLD):** It give the architecture of software product.
- **Low level design (LLD):** It describe how each and every feature in the product should work and every component.

## Implementation:

This phase produces the software under development.

Depending on the methodology, this phase may be conducted in time-boxed “sprints,” (Agile) or may proceed as a single block of effort (Waterfall.)

Regardless of methodology, development teams should produce working software as quickly as possible.

Business stakeholders should be engaged regularly, to ensure that their expectations are being met.

The output of this phase is testable, functional software.

## Testing

It is impossible to deliver quality software without testing. There is a wide variety of testing necessary to measure quality:

- Code quality
- Unit testing (functional tests)
- Integration testing
- Performance testing
- Security testing

The best way to ensure that tests are run regularly, and never skipped for expediency, is to *automate* them. Tests can be

automated using Continuous Integration tools, like Codeship, for example.

## Deployment

The deployment phase is, ideally, a highly automated phase. In high-maturity enterprises, this phase is almost invisible; software is deployed the instant it is ready.

Enterprises with lower maturity, or in some highly regulated industries, the process involves some manual approvals.

However, even in those cases it is best for the deployment itself to be fully automated in a **continuous deployment** model.

**Application Release Automation** (ARA) tools are used in medium and large-size enterprises to automate the deployment of applications to Production environments.

ARA systems are usually integrated with Continuous Integration tools. The output of this phase is the release to Production of working software.

## Operations and maintenance

The operations and maintenance phase is the “end of the beginning,” so to speak.

The Software Development *Life Cycle* doesn’t end here.

Software must be monitored constantly to ensure proper operation.

Bugs and defects discovered in Production must be reported and responded to, which often feeds work back into the process.

Bug fixes may not flow through the entire cycle, however, at least an shortened process is necessary to ensure that the fix does not introduce other problems (known as a regression.)

## Tools Information as per SDLC

- **Requirement Tools:**
  - IBM Requisite Pro - By IBM
  - Cradle - Systems engineering and requirements management tool by 3SL.
  - Dimensions RM - Requirements Management Tool by Serena Software.
  - Concerto - Requirements management platform by Parasoft.
  - Analyst Pro - Software Requirements Tool by Goda Software

**Design Tools**

- IBM Rational Rose
- MS Visio
- MS Office
- Argo UML
- Magic Draw
- Rational Software Architect (IBM)

- **Project Management tools**
  - MS Project
  - MS Office
  - Customized Project Management tracking tools - for metrics
- **Configuration Management Tools**
  - IBM ClearCase
  - VSS
  - SVN
  - AccuRev
  - Git

- **Java development tools (IDE)**
  - IBM's VisualAge for Java development
  - Borland's Jbuilder for Java development
  - Eclipse is from IBM
  - Nebeans is from Sun/oracle
  - Spring source tool (STS)
  - IntelliJ from Idea
  - Rational Software Architect (IBM)
- **Code Review Tools**
  - Code Collaborator
  - Cast
  - Eclipse
  - Sonar
- **Microsoft development tools (IDE)**
  - Visual Studio
- **Application Servers**
  - Jboss
  - Glassfish (Open Source)
  - Apache Geronimo Applications Server

- IBM - WebSphere Application Server
- BEA WebLogic Server

- **Web server**
  - Apache Web Server (65% applications are hosted on apache)
  - Internet Information Server (IIS) - 15% applications are hosted on IIS.
  - Nginx from Nginx hosts around 12% of applications.
  - GWS from Google (5%)
  - Zeus Web Server
  - iPlanet Web Server
  - Roxen Web Server
  - Jigsaw
  - JRun
  - Sambar Server
  - Sun Java System Web Server
- **Portal Server**
  - Websphere portal server
  - Oracle WebCenter

- Adobe LifeCycle

- **Middleware tools**
  - Mule ESB
  - IBM Tivoli
  - Snaplogic
  - Dell Boomi
- **SOA Based tools**
  - SalesForce
  - Amazon Web Services (AWS)
  - Windows Azure
  - Google
- **BPM Modeling tools**
  - JBPM
  - Activiti
  - WPS Websphere process server
- **Build Tools**
  - Apache Maven is the build engine
  - Rake
  - Ant +Ivy

- Gradle
- Gant
- Buildr

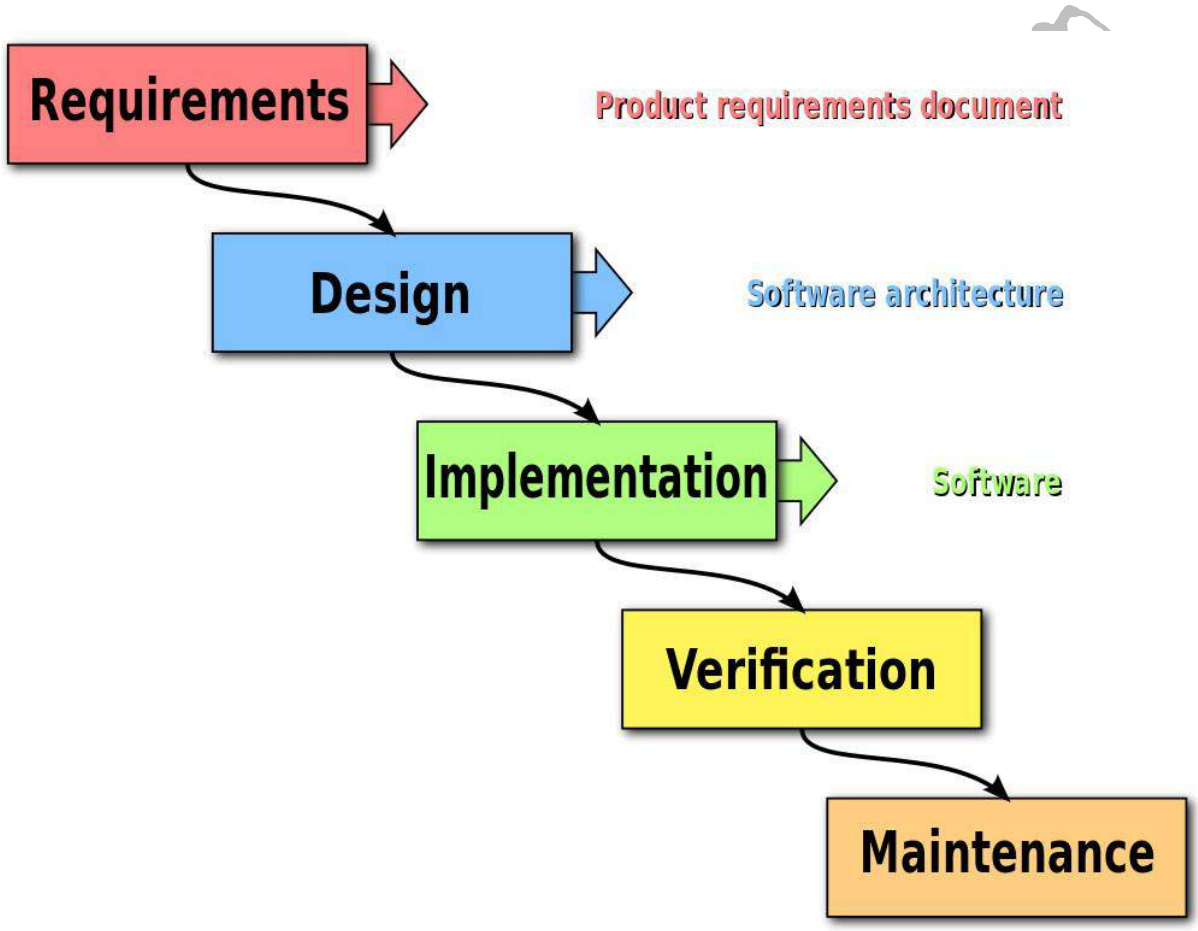
- **Agile Project Management Tools**
  - Rally
  - Jira Agile
  - Redmine
  - Extreme Planner
  - VersionOne
  - TargetProcess
  - AgileTrack
  - Bamboo
- **Testing Tools**
  - QTP/UFT
  - RFT
  - Test Manager
  - Selenium
  - Concordian
  - JMEter

- Jprofiler
- Jprobe
- Loadrunner

- **Defect Tracking Tools**
  - JIRA
  - ClearQuest
  - Rally - Online service for agile project management, requirements and test case management, and defect tracking. Consulting for agile project management. Located in Boulder, Colorado, USA
  - BugZilla
- **Continuous Integration**
  - Hudson
  - Cruise Control
  - TFS (MS First Sight)
- **Hot deployments**
  - JRebel
- **Collaboration tools**
  - Zipline,

- Basecamp
- Sharepoint

## Waterfall Model




This method was adapted from traditional engineering.

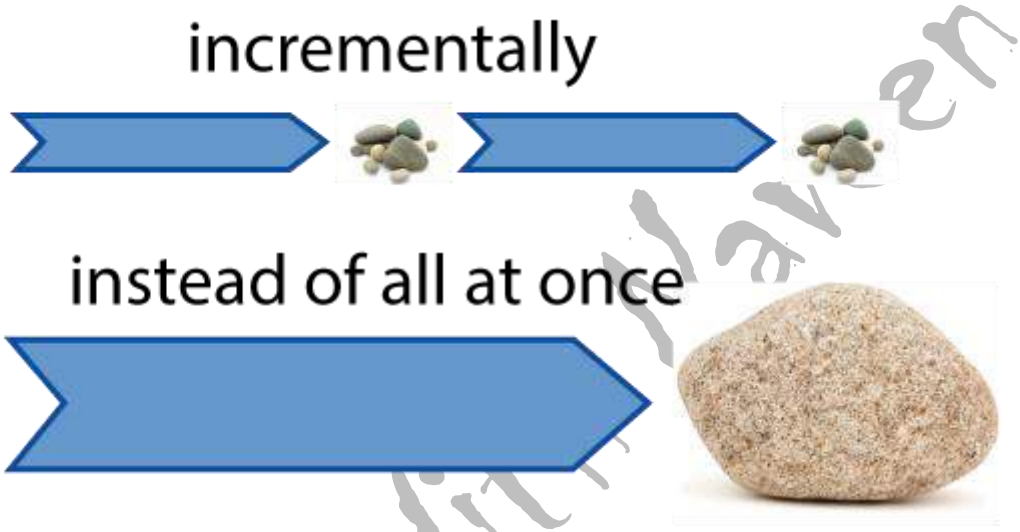
Ironically, the paper credited as the origin of the Waterfall method, describes it as being fundamentally flawed. The method that is known today as “Waterfall” was mistakenly derived from a misunderstanding of this original work. Despite that fact, Waterfall became a very common,

even standard methodology for large projects around the world.

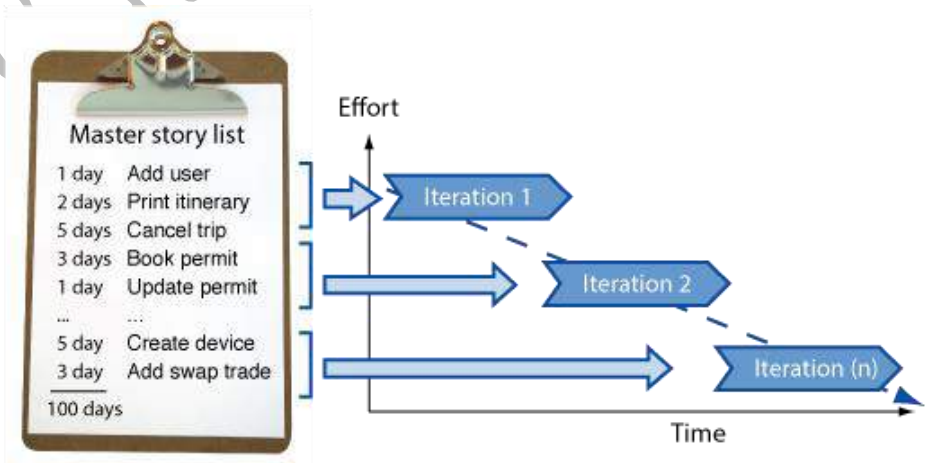
Waterfall methodology begins with long planning and design phases. Once developed, the software then goes through phases of testing, and is finally deployed for use. Waterfall is considered by many to be too rigid to adapt to changing requirements. It does not support feedback throughout the process, leading to the implementation of requirements that may have changed during the development effort. This weakness in Waterfall led to the development of more flexible methodologies, such as Agile.

# What is Agile

Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.




It works by breaking projects down into little bits of user functionality called user stories, Compute them, and then continuously delivering them in short two week cycles called iterations.



# How does it work?

## You make a list

Sitting down with your customer you make a list of features they would like to see in their software. We call these things user stories and they become the To Do list for your project.


## You size things up

Then, using Agile estimation techniques, you size your stories relatively to each other, coming up with a guess as to how long you think each user story will take.

## You set some priorities

Like most lists, there always seems to be more to do than time allows. So you ask your customer to setup their list so you get the most important stuff done first, and save the least important for last.


## You start executing

Then you start delivering some value. You start at the top. Work your way to the bottom. Building, iterating, and getting feedback from your customer as you go.

**You update the plan as you go.**

Then, as you and your customer starting delivering, one of two things is going to happen. You'll discover:

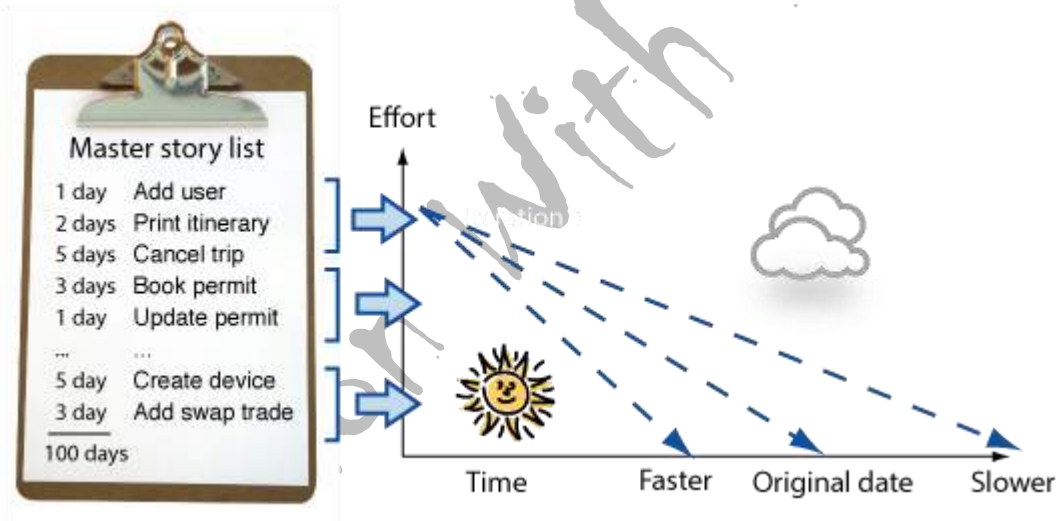
a. You're going fast enough. All is good. Or,

b. You have too much to do and not enough time.

At this point you have two choices. You can either

a) do less and cut scope (recommended). Or you can

b) push out the date and ask for more money.

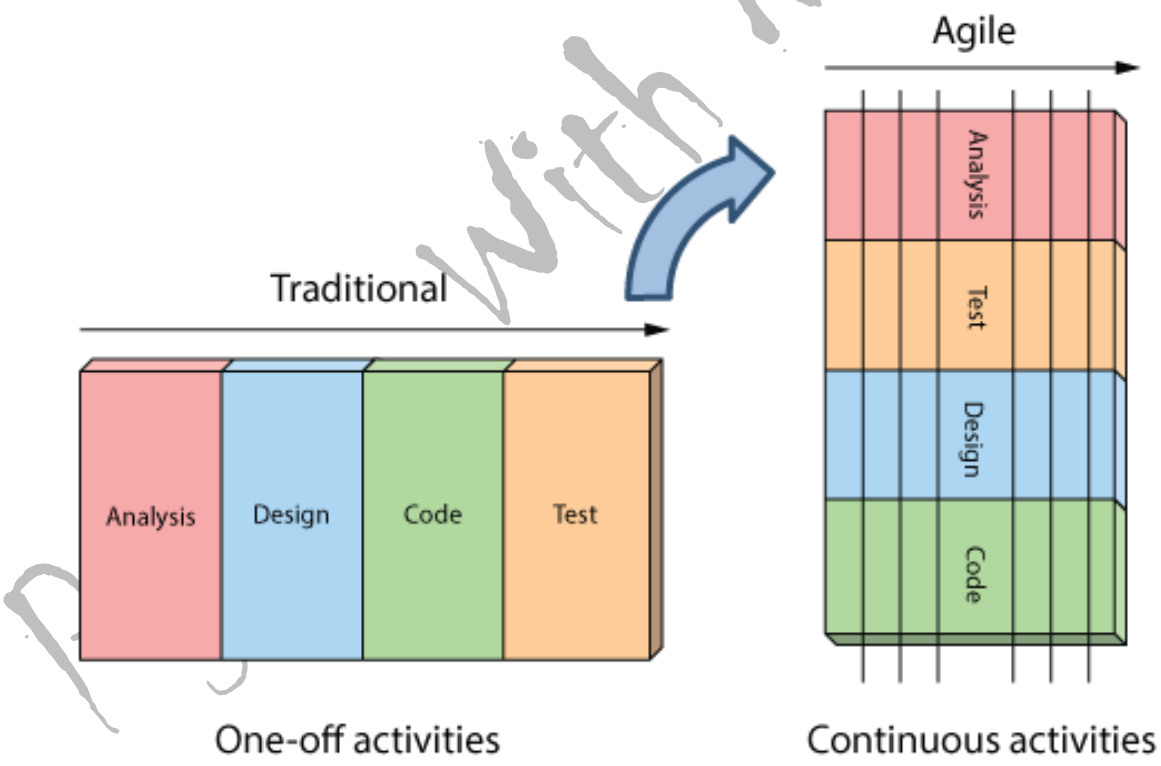


# How is Agile different?

**Analysis, design, coding, and testing are continuous activities**

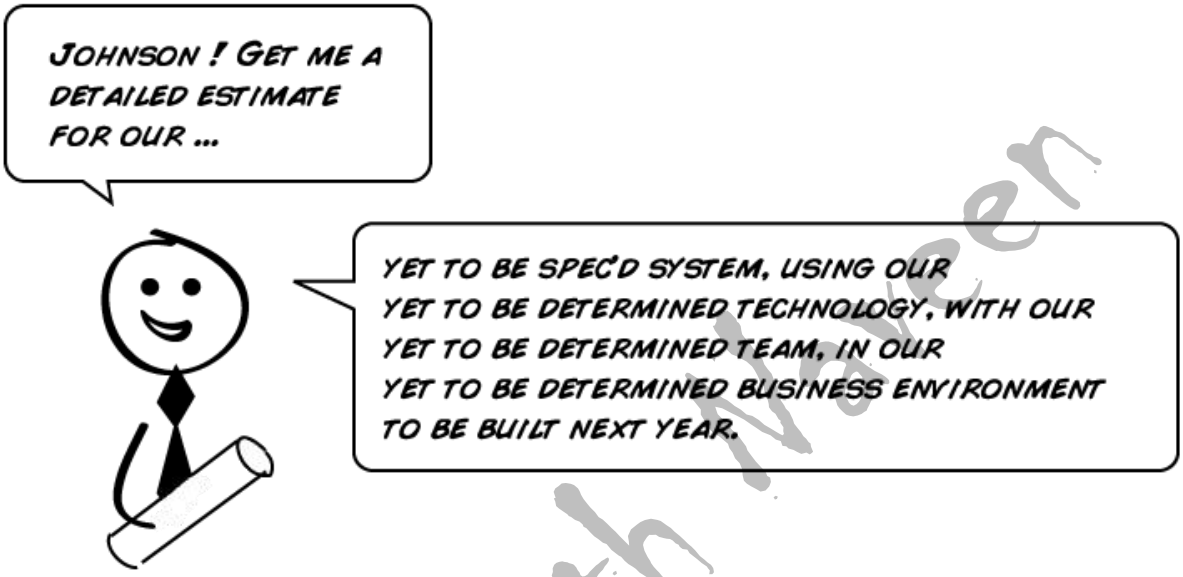
You are never done analysis, design, coding and testing on an Agile project.

So long as there are features to build, and the means to deliver them, these activities continue for the duration of the project.



# Estimation

The fine art of expectation guessing




While we aren't very good at estimating things absolutely, it turns out we are pretty good at estimating things relatively.

Sizing stories relatively means not worrying about *exactly* how big a story is, and worrying more how this story's size compares to others.


This style of estimation (relative over absolute) forms the corner stone of Agile Planning. By sizing our stories relatively, and feeding actual back into our plan, we can make some really accurate predictions about the future while based on what we've done in the past.

To learn more watch this short video on Agile estimation.

# Interview Questions and Answers

## What are the burn-up and burn-down chart?

The burn-up chart depicts the amount of work done in the project, whereas the burn-down chart illustrates the amount of work remaining in the project.

Thus, the burn-up and burn-down are used to describe the progress report of the project.

## What is Daily Stand-Up?

The daily stand-up is the day-to-day meeting (mostly in the morning) in which the whole team meets around 15 minutes to find the answer for the following three questions:

- What was done yesterday?
- What is your plan for today?
- Is there any problem that restricts you to complete your task?

## What is Scrum?

**Scrum is a framework** that helps agile teams work together to develop, deliver, and sustain the complex product in the shortest time. The product provides by scrum team in this shortest period is known as a **sprint**.

## What are the different roles in Scrum?

There are three different roles in scrum. These are the *Scrum Master*, *Product Owner*, *Agile Development Team*:

- **Scrum Master:** The Scrum Master is a team leader and facility provider who help the team member to follow agile practices so that they can meet their commitments and customers requirements.
- **Product Owner:** The Product Owner is one who runs the product from a business perspective. He defines the requirements and prioritizes their values.
- **Agile Development Team:** Agile development team provides the judgment on the technical feasibilities or any dependencies.

## What Scrum Master do?

- Tracking and monitoring project development.
- Understanding the user requirement correctly.
- Work to obtain the project properly.
- Improving the performance of the team.
- Organized meetings and resolve issues.
- Communicate and report to the customer and development team.

## What is Scrumban?

The Scrum and Kanban as flavors of Agile.

Scrum is best-suited for products and development projects.

Kanban is best for production support.

Scrumban – which combines the best features of both – for maintenance projects.

Scrumban is becoming very popular these days in service industries, where we have both development and maintenance projects.

## What is agile testing?

The agile testing is the software testing process which is fully based on the principle of agile software development. It is the iterative approach where the user story becomes the output of the collaboration between the product owner and the development team.

## What are the agile frameworks.

- Scrum
- Kanban
- Feature Driven Development
- Test Driven Development

## What is your project sprint length ?

**Note :**

**1) It is a common question for experienced people.**

**2) The idea behind is to judge in which kind of environment you have worked**

There will be follow up of the question that the length fixed in the beginning and never changed? Did you try with less than this length or more than that?