10/11/2022

DEFECT CATEGORIES

**Definition: Defects** are defined as the deviation of the actual and expected result of system or software application. Defects can also be defined as any deviation or irregularity from the specifications mentioned in the product functional specification document.

**Types of Defects:**

the defects made by the developer in some arithmetic expression or mistake

the bugs deal with problems related toUI are usually considered less severe.

These defects affect the functionality of the application - All JavaScript errors

The occurrence of these bugs hampers the crucial functionality of the application

Logical defects are mistakes done regarding the implementation of the code.

system or the software application is unable to meet the desired and the expected results

Interface defects means the defects in the interaction of the software and the users.

Multithreading means running or executing the multiple tasks at the same time.

Syntax defects means mistake in the writing style of the code.

improper handling of data sent from the user to the application.

Deals with improper handling of data in the database

DEFECT

11/11/2022

**Severity and Priority**

**1)High Severity Low Priority(HSLP)**

 Eg. If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the **severity is high but priority is low**.

**2)Low Severity High Priority(LSHP)**

If the company name is misspelled in the home page of the website,then the **severity is low and priority is high** to fix it.

**3)High Priority & High Severity(HPHS)**

Eg. A site maintaining the student details, on saving record if it, doesn't allow to save the record then this is **high priority and high severity.**

**4)Low Priority and low severity(LPLS)**

 Eg. If the privacy policy of the website has a spelling mistake, this defect is set as **Low Severity and Low Priority.**

**KANBAN**

Kanban is a visual-driven model for project management that [focuses on continuous improvement](https://www.planview.com/resources/guide/lean-principles-101/what-is-continuous-improvement/) and transparency in workflows. At the simplest level, Kanban in software testing is all about visualizing project status to clearly understand what’s sitting in a backlog waiting to get picked up, what work is currently in progress and what work is complete..

Kanban Cards:

The Kanban cards are essential pieces on the Kanban board as it represents the work that the team is working on. These cards will have

1. Priority
2. Owner
3. Type
4. Due date

A column in Kanban board represents the work stage, and you can place a WIP (Work in Progress) limit on the column. ***The* WIP limit means the maximum number of cards that can stay on that column**.

Since Kanban project management uses a pull-based system, as and when a developer is free, he/she can pull a card from the to-do column to the dev column.

**Kanban Board**

**Kanban Board** is an agile project management tool that helps implement Kanban to manage projects for personal and business purposes. It is a physical or digital (JIRA) board designed to help teams visualize their work at different stages and processes. It also helps represent the stages of work with columns using cards.

It has columns that represent the status of the work like

1. To-do,
2. Dev
3. Testing
4. Done.

Each of these columns can have cards <=the WIP limit. The cards represent the actual work.

You can use positive numbers to limit work-in-progress, and this limit number can be placed on the top of the columns in both physical and digital Kanban boards. Any individual of the team can manage the state of his card, and the entire team can visualize the workflow. Next in this Kanban tutorial, we will learn about Kanban Workflow.

**Kanban Workflow**

**Kanban Workflow** is a set of steps that helps teams to define explicit policies and principles in Kanban. It represents the rules and procedures while the work is going on across various stages of development and delivery cycles. Kanban workflow consists of step-by-step processes between starting and the delivery of a particular task.

The basic principal Kanban follows is, **“stop starting, start finishing”.**With the help of WIP limits, it gets more work done. There are customizable Kanban workflows and states available in any modern tool like JIRA.

Below are the basic states that many software teams follow for their workflow management.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**The Four Principles of Kanban**

Below are the main Four core principles of Kanban:

1. **Start with what you have now**: Kanban system suggests working incrementally and start with what you have currently. Since one of its practice is to improve continuously, you must improve the system gradually.
2. **Agree to Pursue Incremental, Evolutionary Change:** Kanban recommends an incremental change in the process, and you must not make a big change in the process in one go.
3. **Respect the Current Process, Roles & Responsibilities:** Once again, start with what you have now and change the process, role, and responsibilities in an incremental manner.
4. **Encourage Acts of Leadership at All Levels**: Every individual can act as a leader and provide ideas to improve the efficiency of the overall Kanban system. You should not think that this is a management level activity, and even the youngest member of the team can act as a leader.

**The Six Kanban Core Practices**

Following are the main Six core practices of Kanban:

1. **Visualize the workflow**: This principle suggests having a Kanban board (physical or digital) to visualize the workflow. Each individual of a team must see his card and cards of other team members. You can move your cards in different columns as per the above image. It brings lots of transparency within the team and also makes it easier to resolve blockers
2. **Limit work in progress**: Kanban is a pull-based system, and it improves the efficiency of a team to limit work in progress and have tasks that can be completed in the given time frame by the team. This WIP limit applies from the beginning to the end of the workflow. You can apply the limit on top of the column using a positive integer.
3. **Focus on flow**: This principle focuses on flow and on any interruptions. If there are interruptions or blockers, they must be fixed permanently.
4. **Explicit Policies**: Policies can be established in a team to reduce the rework and focus on the areas which require attention or where it is more effective.
5. **Feedback Loop**: Feedback loops are very essential in Kanban. It is not just within the team but between multiple teams, coaches, etc. This helps in improving the overall health of the Kanban system.
6. **Continuous Improvement**: This is the core principle of the Kanban system. It states that you can always improve the process, and that will result in better efficiency.

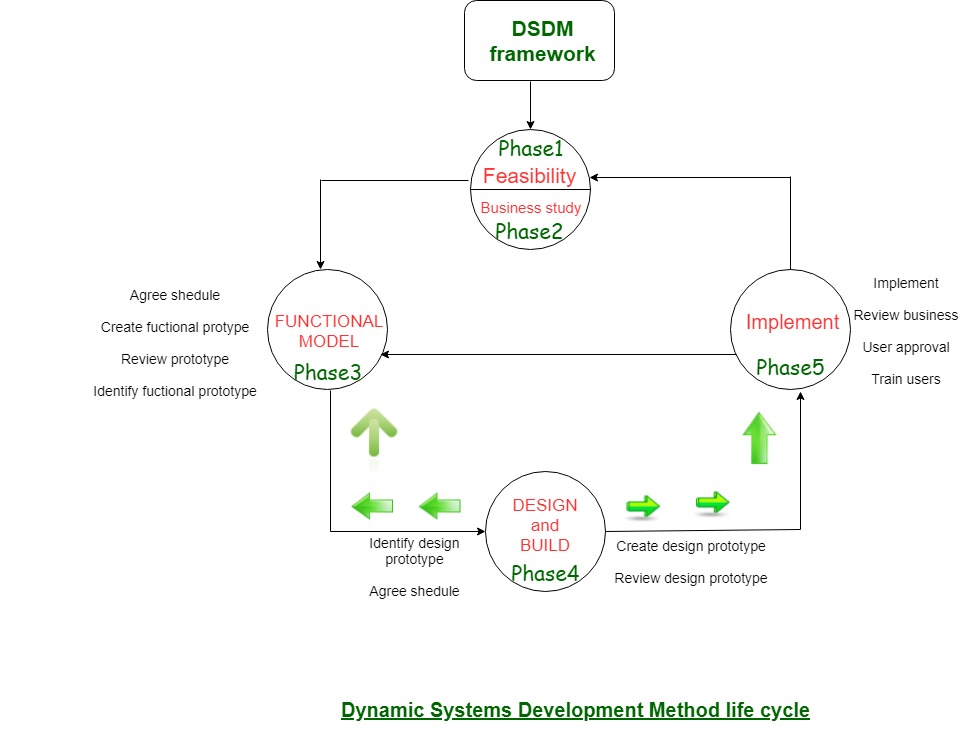
The **Dynamic Systems Development technique (DSDM)** is an associate degree agile code development approach that provides a framework for building and maintaining systems. The DSDM philosophy is borrowed from a modified version of the sociologist principle—80 % of An application is often delivered in twenty percent of the time it’d desire deliver the entire (100 percent) application.

DSDM is An iterative code method within which every iteration follows the 80% rule that simply enough work is needed for every increment to facilitate movement to the following increment. The remaining detail is often completed later once a lot of business necessities are noted or changes are requested and accommodated.

The DSDM tool (www.dsdm.org) could be a worldwide cluster of member companies that put together tackle the role of “keeper” of the strategy. The pool has outlined AN [Agile Development Model](https://www.geeksforgeeks.org/software-engineering-agile-development-models/), known as the DSDM life cycle that defines 3 different unvarying cycles, preceded by 2 further life cycle activities:

1. **Feasibility Study:**  
   It establishes the essential business necessities and constraints related to the applying to be designed then assesses whether or not the application could be a viable candidate for the DSDM method.
2. **Business Study:**  
   It establishes the use and knowledge necessities that may permit the applying to supply business value; additionally, it is the essential application design and identifies the maintainability necessities for the applying.
3. **Functional Model Iteration:**  
   It produces a collection of progressive prototypes that demonstrate practicality for the client.  
   (Note: All DSDM prototypes are supposed to evolve into the deliverable application.) The intent throughout this unvarying cycle is to collect further necessities by eliciting feedback from users as they exercise the paradigm.
4. **Design and Build Iteration:**  
   It revisits prototypes designed throughout useful model iteration to make sure that everyone has been designed during a manner that may alter it to supply operational business price for finish users. In some cases, useful model iteration and style and build iteration occur at the same time.
5. **Implementation:**  
   It places the newest code increment (an “operationalized” prototype) into the operational surroundings. It ought to be noted that:
   * **(a)** the increment might not 100% complete or,
   * **(b)** changes are also requested because the increment is placed into place. In either case, DSDM development work continues by returning to the useful model iteration activity.

Below diagram describe the DSDM life cycle:



DSDM is often combined with XP to supply a mixed approach that defines a solid method model (the DSDM life cycle) with the barmy and bolt practices (XP) that are needed to create code increments. additionally, the ASD ideas of collaboration and self-organizing groups are often tailored to a combined method model.

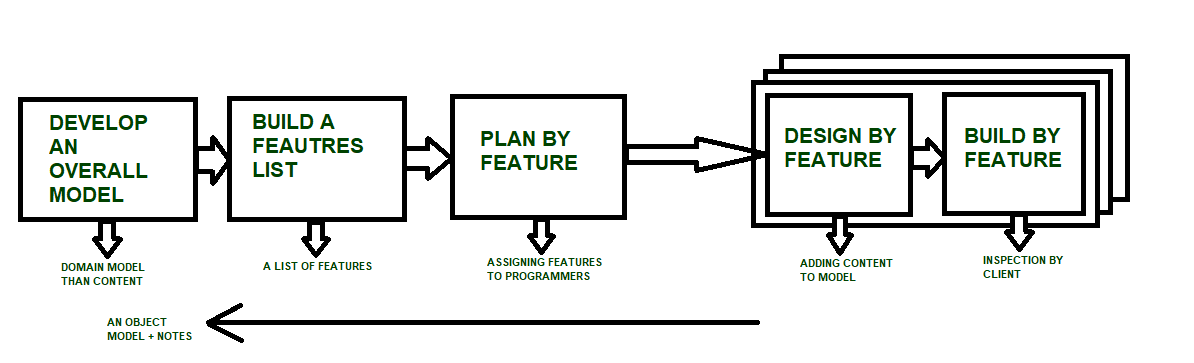
[Agile Methodology: What is Agile Model in Software Testing? (guru99.com)](https://www.guru99.com/agile-scrum-extreme-testing.html)

FDD

**FDD** stands for **Feature-Driven Development**. It is an agile iterative and incremental model that focuses on progressing the features of the developing software. The main motive os feature-driven development is to provide timely updated and working software to the client. In FDD, reporting and progress tracking is necessary at all levels.

#### FDD Lifecycle

* Build overall model
* Build feature list
* Plan by feature
* Design by feature
* Build by feature



#### Characteristics of FDD

* **Short iterative:** FDD lifecycle works in simple and short iterations to efficiently finish the work on time and gives good pace for large projects.
* **Customer focused:** This agile practice is totally based on inspection of each feature by client and then pushed to main build code.
* **Structured and feature focused:** Initial activities in lifecycle builds the domain model and features list in the beginning of timeline and more than 70% of efforts are given to last 2 activities.
* **Frequent releases:** Feature-driven development provides continuous releases of features in the software and retaining continuous success of the project.

#### Advantages of FDD

* Reporting at all levels leads to easier progress tracking.
* FDD provides continuous success for larger size of teams and projects.
* Reduction in risks is observed as whole model and design is build in smaller segments.
* FDD provides greater accuracy in cost estimation of the project due to feature segmentation.

#### Disadvantages of FDD

* This agile practice is not good for smaller projects.
* There is high dependency on lead programmers, designers and mentors.
* There is lack of documentation which can create an issue afterwards.

## What is OOPS?

**Object-Oriented Programming System (OOPs)** is a programming concept that works on the principles of abstraction, encapsulation, inheritance, and polymorphism.

### 1) Class

The class is one of the Basic concepts of OOPs which is a group of similar entities. It is only a logical component and not the physical entity.

Example: if you had a class called “Expensive Cars” it could have objects like Mercedes, BMW, Toyota, etc

### 2) Object

An object can be defined as an instance of a class, and there can be multiple instances of a class in a program. An Object is one of the Java OOPs concepts which contains both the data and the function, which operates on the data.

Example – chair, bike, marker, pen, table, car, etc.

### 3) Inheritance

Inheritance is one of the Basic Concepts of OOPs in which one object acquires the properties and behaviors of the parent object.

It’s creating a parent-child relationship between two classes. It offers robust and natural mechanism for organizing and structure of any software.

### 4) Polymorphism

Polymorphism refers to one of the OOPs concepts in Java which is the ability of a variable, object or function to take on multiple forms.

Example, in English, the verb *run* has a different meaning if you use it with *a laptop*, *a foot race*, and *business*. Here, we understand the meaning of *run* based on the other words used along with it. The same also applied to Polymorphism.

### 5) Abstraction

Abstraction is one of the OOP Concepts in Java which is an act of representing essential features without including background details. It is a technique of creating a new data type that is suited for a specific application.

Example, while driving a car, you do not have to be concerned with its internal working. Here you just need to concern about parts like steering wheel, Gears, accelerator, etc.

### 6) Encapsulation

Encapsulation is one of the best Java OOPs concepts of wrapping the data and code. In this OOPs concept, the variables of a class are always hidden from other classes. It can only be accessed using the methods of their current class.

Example – in school, a student cannot exist without a class.

### 7) Association

Association is a relationship between two objects. It is one of the OOP Concepts in Java which defines the diversity between objects. In this OOP concept, all objects have their separate lifecycle, and there is no owner.

Example-many students can associate with one teacher while one student can also associate with multiple teachers.

### 8) Aggregation

In this technique, all objects have their separate lifecycle. However, there is ownership such that child object can’t belong to another parent object.

Example- consider class/objects department and teacher. Here, a single teacher can’t belong to multiple departments, but even if we delete the department, the teacher object will never be destroyed.

### 9) Composition

Composition is a specialized form of Aggregation. It is also called “death” relationship. Child objects do not have their lifecycle so when parent object deletes all child object will also delete automatically.

Example of House and rooms. Any house can have several rooms. One room can’t become part of two different houses. So, if you delete the house room will also be deleted.

|  |  |
| --- | --- |
| **For Loop** | **While Loop** |
| It is used when the number of iterations is known. | It is used when the number of iterations is not known. |
| In case of no condition, the loop is repeated infinite times. | In case of no condition, an error will be shown. |
| Initialization is not repeated. | Initialization is repeated if carried out during the stage of checking. |
| Statement of Iteration is written after running. | It can be written at any place. |
| Initialization can be in or out of the loop | Initialization is always out of the loop. |
| The nature of the increment is simple. | The nature of the increment is complex. |