27.11.2019

Extreme Programmimg And Service Oriented Architecture

Subject - System Analysis and Design
Teacher - Mr. Sunil Sharma
By - Dipesh Gupta & Manoj Sitoula

What is software?

Simply, Software is a set of instructions that tells the computer about the tasks to be performed and how these tasks are to be performed.

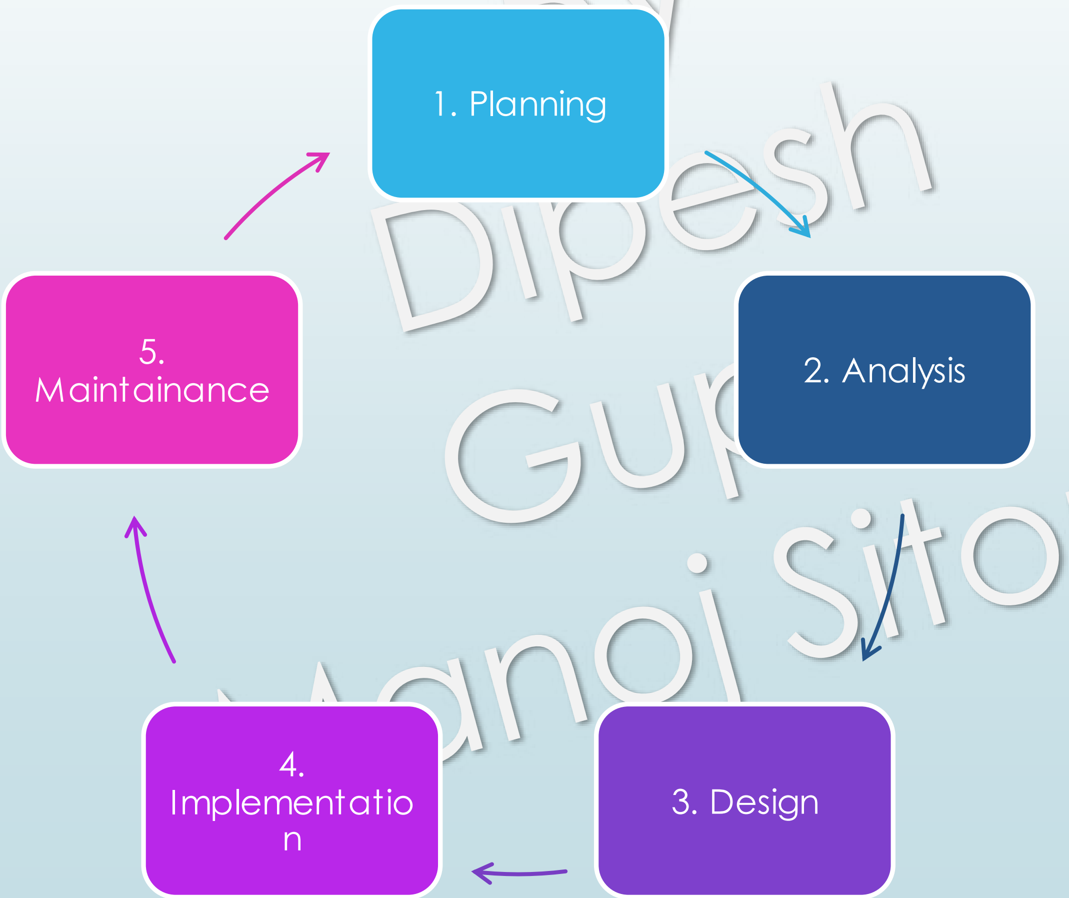
In another words, Software is defined as a collection of programs, documentation and operating procedures.

Types of software are:

- **System software**
- **Real-time software**
- **Business software**
- **Engineering and scientific software**
- **Artificial intelligence (AI) software**
- **Web-based software**
- **Personal computer (PC) software**

Software Development Life Cycle (SDLC)

SDLC is a common methodology for the development, maintainance and replacing/upgrading of system used by many organization.



Types of Software Development Models

- Traditional Waterfall Model.
- Prototyping Model.
- Rapid Application Development Model.
- Incremental Model.
- Spiral Model.
- Agile Model.
- Extreme Programming Model.
- Service Oriented Architecture Model. etc

Extreme programming(XP)

- Extreme programming is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements.
- Extreme Programming (XP) is one of the numerous Agile frameworks applied by IT companies.
- Extreme programming includes programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, code simplicity and clarity, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers.
- The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels.

Values of Extreme programming.

- **Fine-scale feedback**
  - Pair programming
  - Planning game
  - Test-driven development
  - Whole team

- **Continuous process**
  - Continuous integration
  - Refactoring or design improvement
  - Small releases

- **Shared understanding**
  - Coding standards
  - Collective code ownership
  - Simple design
  - System metaphor

- **Programmer welfare**
  - Sustainable pace

Refactoring

When implementing a feature, the developers always ask if there is a way of changing the existing code to make adding the feature simple. After they have added a feature, the developers ask if they now can see how to make the code simpler, while still running all of the tests. They restructure the system without changing its behavior to remove duplication, improve communication, simplify, or add flexibility. This is called Refactoring.

Pair Programming

In Pair programing, the entire code is written with two developers at one machine, with one keyboard and one mouse.

There are two roles in each pair −

•The first developer (the one with the keyboard and the mouse) thinks about the best way to implement this method right here.

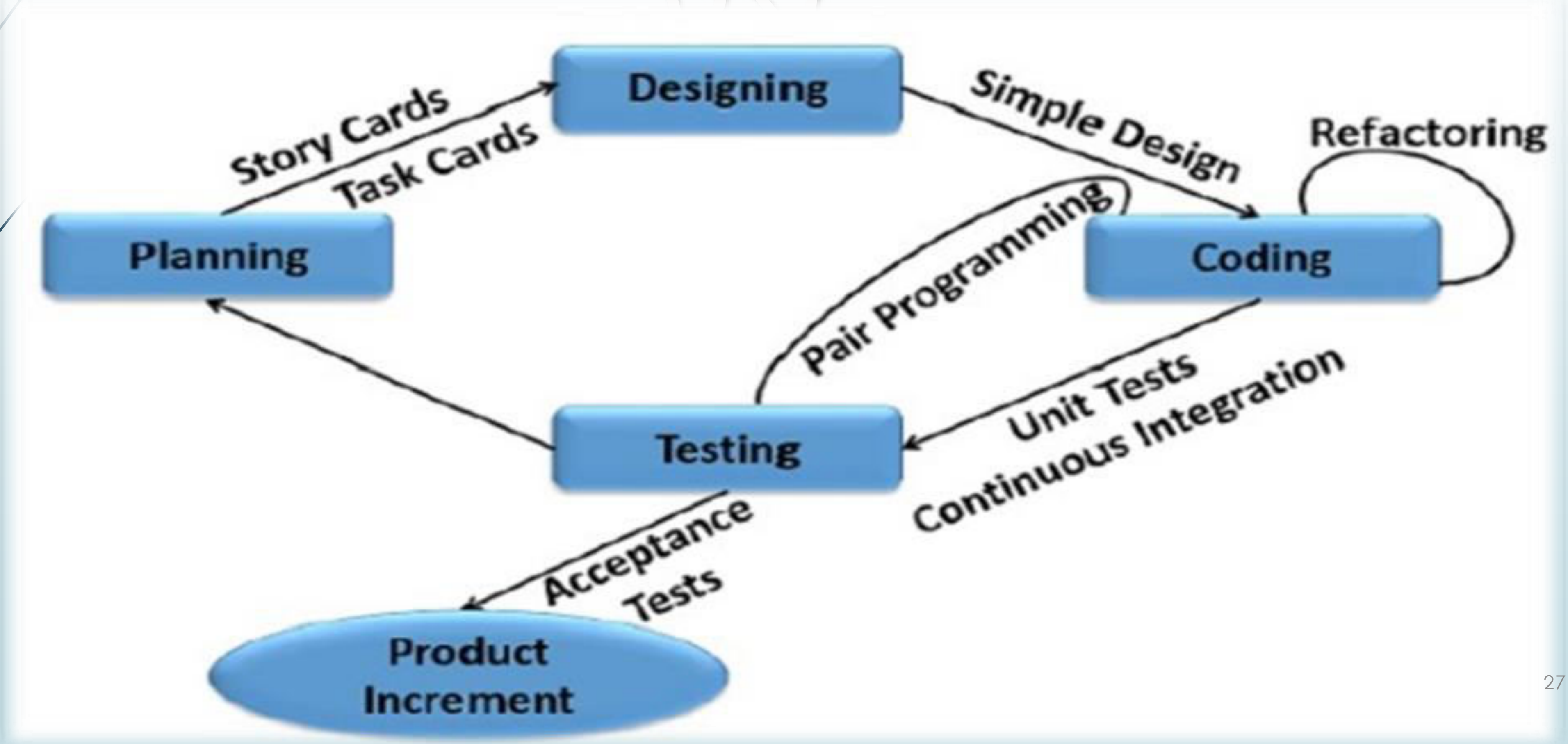
•The other developer is thinks more strategically

- Is this whole approach going to work?
- What are some other test cases that might not work yet?
- Is there some way to simplify the whole system so the current problem just disappears?

The pairing is dynamic. It means that the two Roles A and B may exchange their places, or they may pair up with other team members. More often, anyone on the team will do as a partner. For example, if you have a responsibility for a task in an area that is unfamiliar to you, you might ask someone with recent experience to pair with you.

Life cycle of XP model.

Life cycle of extreme programming model can be shown below :



Advantages and Disadvantages of XP

Advantages

- The main advantage of Extreme Programming is that it save costs and time required for project realization.
- Simplicity is one more advantage of Extreme Programming projects. The whole process in XP is visible and accountable. Developers commit what they will accomplish and show progress.
- Constant feedback is also the strong side.It is necessary to listen and make any changes needed in time.
- XP assists to create software faster thanks to the regular testing at the development stage.
- Extreme Programming contributes increasing employee satisfaction and retention.

Disadvantages

- In XP projects the defect documentation is not always good. Lack of defect documentation may lead to the occurrence of similar bugs in the future.
- One more disadvantage of XP is that this methodology does not measure code quality assurance. It may cause defects in the initial code.
- XP is not the best option if programmers are separated geographically.

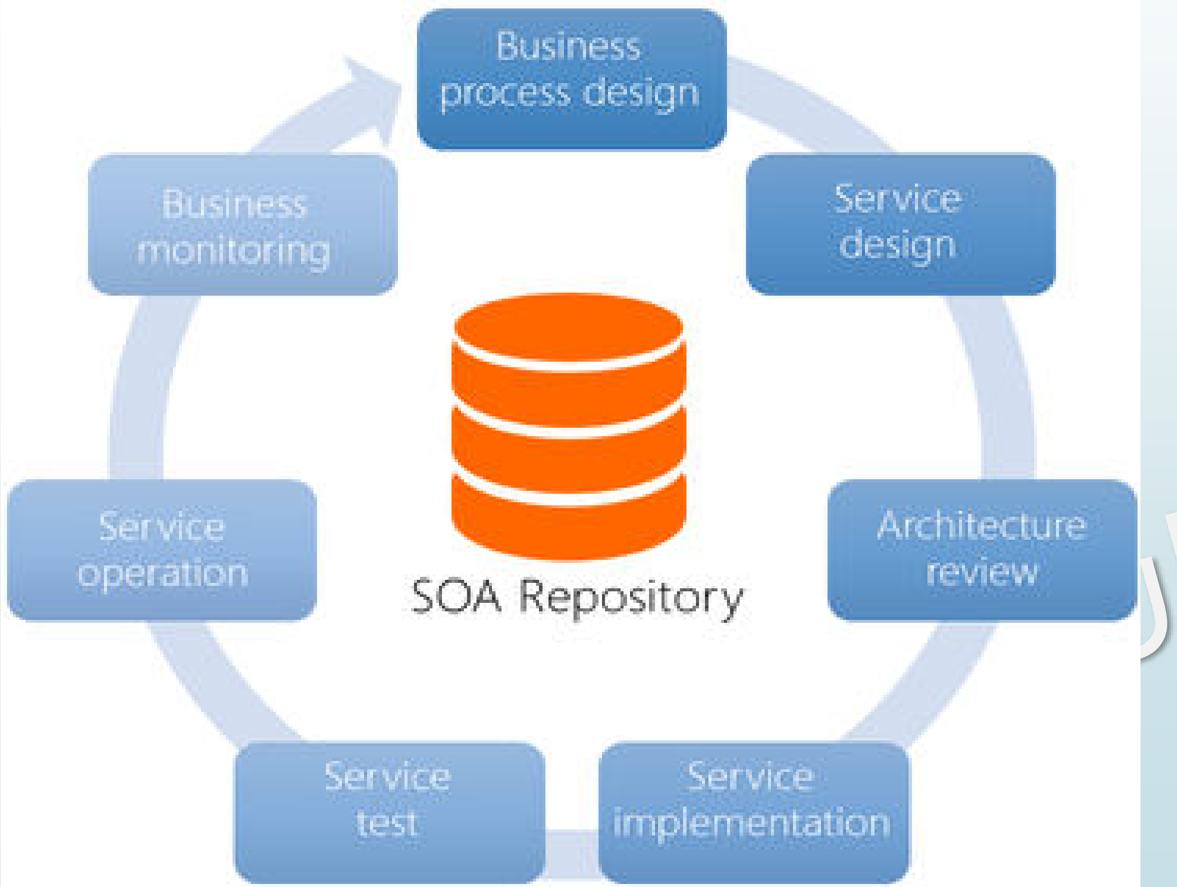
Where to use Extreme Programming

- **Highly-adaptive development.** XP was designed to help development teams adapt to fast-changing requirements.
- **Risky projects.** Teams applying XP practices are more likely to avoid problems connected with working on a new system, especially when a product owner sets strict deadlines for a project.
- **Small teams.** XP practices are efficient for teams that don’t exceed 12 people.
- **Automated testing.** Another factor that can influence the choice of XP is the developers’ ability to create and run unit tests.
- **Available customer participation.** As XP requires customers, developers and managers work side-by-side, make sure your client can spend time in the office until a project ends.

Service Oriented Architecture

- Service Oriented Architecture (SOA) is an architectural approach in which applications make use of services available in the network.
- In this architecture services are provided to form applications through a communication call over the internet.
- SOA allows users to combine a large number of facilities from existing services to form applications.
- SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
- SOA based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.

Service Oriented Architecture(SOA) Working Diagram



Principles of SOA:

- **Standardized Service Contract:**
  Services adhere to a service description. A service must have some sort of description which describes what the service is about. This makes it easier for client applications to understand what the service does.
- **Loose Coupling:**
  Less dependency on each other. This is one of the main characteristics of web services which just states that there should be as less dependency as possible between the web services and the client invoking the web service. So if the service functionality changes at any point in time, it should not break the client application or stop it from working.
- **Service Abstraction:**
  Services hide the logic they encapsulate from the outside world. The service should not expose how it executes its functionality; it should just tell the client application on what it does and not on how it does it.

Principles of SOA:

- **Service Reusability:**
  Logic is divided into services with the intent of maximizing reuse. . In any development company re-usability is a big topic because obviously one wouldn't want to spend time and effort building the same code again and again across multiple applications which require them. Hence, once the code for a web service is written it should have the ability work with various application types.

- **Service Autonomy:**
  Services should have control over the logic they encapsulate. The service knows everything on what functionality it offers and hence should also have complete control over the code it contains.

Principles of SOA:

- **Service Discoverability:**
  Services can be discovered (usually in a service registry). We have already seen this in the concept of the UDDI, which performs a registry which can hold information about the web service.
- **Service Interoperability:**
  Services should use standards that allow diverse subscribers to use the service. This is considered so obvious these days that it is often dropped as a principle.

Advantages of SOA:

- **Service reusability:**

  Because the services provided or requested are complete, self-contained programs, they are characterized by their re-usability. Someone has already created the scissors. Why would someone else try to create their own if they needed to cut fabric? Services are re-used many times in different applications irrespective of the other interacting components in that system. Let's take a currency converter for example. Not only do financial websites use them, but popular blogs and websites add these services to make their content diversified, giving the user space to do different things.

- **Platform independent:**

  One of the main advantages of SOA services is their public availability to any system that needs them, independent of the platform on which the system runs. This platform independence allows the integration of different requested services from different vendors to run harmoniously, regardless of the platform.

- **Availability:**

  Normally, services in a SOA are available to any requester that needs them. For this reason, SOA services are made available via public directories where they can be searched for and discovered. Consumers, therefore, are always able to access the

Advantages of SOA:

- **Reliability:**

SOA services are complete and self-contained programs. This makes it easy for testing, debugging or any form of maintenance.

- **Scalability:**

Different organizations have different levels of complexity in their architecture. So, the ability of services to be successfully run on different servers within an environment increases the scalability and availability of the service.

- **High-Quality Services:**

Servers, as we have seen, may have many applications to which they may be applied. As such, there will be no functional redundancy. Data is always consistent and the service codes are constantly maintained resulting in continuous high-quality applications.

Disadvantages of SOA:

- **High overhead:**

  Every time a service interacts with another service, complete validation of every input parameter takes place. This increases the response time and machine load, and thereby reduces the overall performance.

- **High investment:**

  Implementation of SOA requires a large upfront investment by means of technology, development, and human resource.

- **Complex service management :**

  The service needs to ensure that messages have been delivered in a timely manner. But as services keep exchanging messages to perform tasks, the number of these messages can go into millions even for a single application. This poses a big challenge to manage such a huge population of services.

The End

By Dipesh Gupta
Manoj Sitoula