**Software Development**

Table of Contents

[Introduction 2](#_Toc65540157)

[Waterfall Model 4](#_Toc65540158)

[Prototyping Model 5](#_Toc65540159)

[Agile Methodologies 6](#_Toc65540160)

[Changes from Agile Process 7](#_Toc65540161)

## Introduction

In software engineering, a software development process is the process of dividing software development work into distinct phases to improve design, product management and project management. It consists of the following steps:

* Communication
* Planning
* Analysis
* Design
* Coding
* Deployment
* Maintenance

Communication is the process of figuring out what the requirements of the customer are. It takes a story, use case or scenario and tries to create a product requirement document from it.

Planning is the creation of the architectural blueprint for the project. It figures out estimates for the budget, cost, time, technology, project management, experience and scope for the project.

Analysis is the details of the architectural blueprint, or simple what to do in the project. It plans out functions, methods, classes, objects, interactions and states, which results in models, schema and business rules

Design is the how to do, or the pseudocode for the project. It consists of packages, class-class interaction, designs, patterns, architectural patterns and modules and results in the software architecture.

Coding consists of around just 20% of the total work of the project. It consists of writing the actual code for the software, systematically testing it in order to discover and debug defects and validating it.

The deployment stage is when the final software is delivered to the customer.

The maintenance stage is the support that must be provided to the customer after delivery, while they are using the software.

## Waterfall Model

The waterfall model is a straight-forward method. It simply follows the steps of the software development process one by one, in order. This means that, for projects that have clearly defined requirements that do not change over time and have well organized teams, it is the best model to follow. However, it is not flexible at all. If any mistakes are made or if there are changes in the requirements, the whole project must basically be redone, which drives up costs. The later in the process an error is found or a change made, the greater the cost associated with it.

There are many other problems associated with the waterfall model. Analysis paralysis is one such problem where exceedingly long phases of project planning, requirements gathering, program design and modelling occur. These do not always create much extra value and risk many revisions, causing delays and increasing costs. Validation problems may also occur if the requirements of the customer are not understood well.

A major problem with this model is at the integration stage. Most large software is actually built in small parts by different teams. Each of these parts are then individually tested (unit testing), and then brought together. Even though the parts may work extremely well individually, they may fail the testing stage when they are brought together, or integrated (integration testing). As the integration stage comes very late in the process in the waterfall model, it often causes major delays and costs.

Note: Another form of testing is called Validation Testing, which is testing that is done after the entire software has been developed.

## Prototyping Model

The prototyping model is best for situations where requirements are not very well understood or may change. Prototyping is about creating prototypes that are incomplete versions of the software program being developed. It is done to understand customer requirements by giving them a visual aid, allowing them to better describe what exactly they want. It is an example of test-driven development (TDD) and works by quickly going through the initial stages of the software development process several times, relying on backlogs (requirements that have not yet been met) to improve. Customer requirements are taken, some quick planning, design and analysis is done and a small prototype is created and presented to the customer. The customer evaluates the prototype and adjusts requirements and gives suggestions. This process is repeated until the customer is satisfied. After this, the final design, coding and testing are done and the product delivered to the customer.

It is often seen in real life development that customers cannot clearly describe exactly what they want, and may even change their requirements as time goes by. This means that the entire process of software development needs to be flexible to change, be product-centric and give frequent demos to the customers. The prototyping model does exactly this. In fact, it is the result of an agile approach to the process, and is part of a much larger series of improvements that came through agile frameworks, such as continuous integration, pair programming, refactoring, running versions and customer involvement through constant interaction.

## Agile Methodologies

Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software that is responsive to customer requirements, while remaining simple. It also has the shortest cycle among the agile methodologies, lasting only a week and has given rise to many improvements to the development cycle. It relies on communication, simplicity, feedback, courage and respect.

Communication is required to transfer knowledge between team members. XP stresses that this occur face to face, with the aid of a white board or drawing mechanism.

Simplicity refers to finding the simplest solution that meets the requirements. XP tries to avoid waste and do only what is absolutely necessary. This keeps the design of the system simple, making it easier to maintain, support and revise. It also means meeting only the requirements that are known.

Feedback is taken from previous efforts, allowing teams to improve and revise their practices and the product.

Courage refers to the courage to point out problems in the process that are reducing effectiveness, and to change the parts that do not work, regardless of how difficult it may be to accept this.

Respect is essential between team members in order to have proper communication and to provide and accept feedback.

Scrum is an agile framework designed for teams of three to nine members, who break their work into actions that can be completed within timeboxed iterations, called sprints, no longer than one month, but usually two weeks. They then tack progress and re-plan in 15 minute timeboxed stand-up meetings, called daily scrums. In daily scrums, developers discuss what they did yesterday, what they will do today and what difficulties they faced. These discussions help them solve the problems. They work together on a large table so they can collaborate and communicate and if someone finds something difficult or new, they can work on it together.

### Changes from Agile Process

Continuous integration is the practice of merging all developed working copies to shared mainline several times a day. Extreme Programming (XP) adopted the concept of continuous integration and promoted integrating more than once per day, perhaps as many as ten times per day. This means that any problems at the integration stage are caught early in the process, and there is a ‘running version’ of the software always at the ready. The tools required for continuous integration are often automated, to save time.

Pair programming is an agile software development technique in which two programmers work together at one work station. One, the driver, writes the code while the other, the observer or navigator, reviews each line of code as it is typed in, in order to catch bugs. They switch roles frequently.

Refactoring is the process of changing a software system in such a way that it does not alter the external behaviour of the code, yet improves its internal structure. This can make the system less complex and more readable.

Spike solution – A spike is a product testing method origination from XP that uses the simplest possible program to explore potential solutions to a specific problem. It is used to determine how much work will be required to solve or work around a software issue. Typically, a ‘spike test’ involves gathering additional information or testing for easily reproduced test cases.