10.4.3 SDLC and Development Facts

Even though you may not be a software developer, most security professionals at some point work with software engineers who develop applications. And just like any other enterprise component, these applications need to be properly secured.

This lesson covers the following topics:

- Waterfall development life cycle model
- Agile development life cycle model
- Coding errors
- Security testing methods

Waterfall Development Life Cycle Model

The most widely used development model is the Waterfall model. It is called this because each step is completed before the next step is begun so that each step flows to the next.

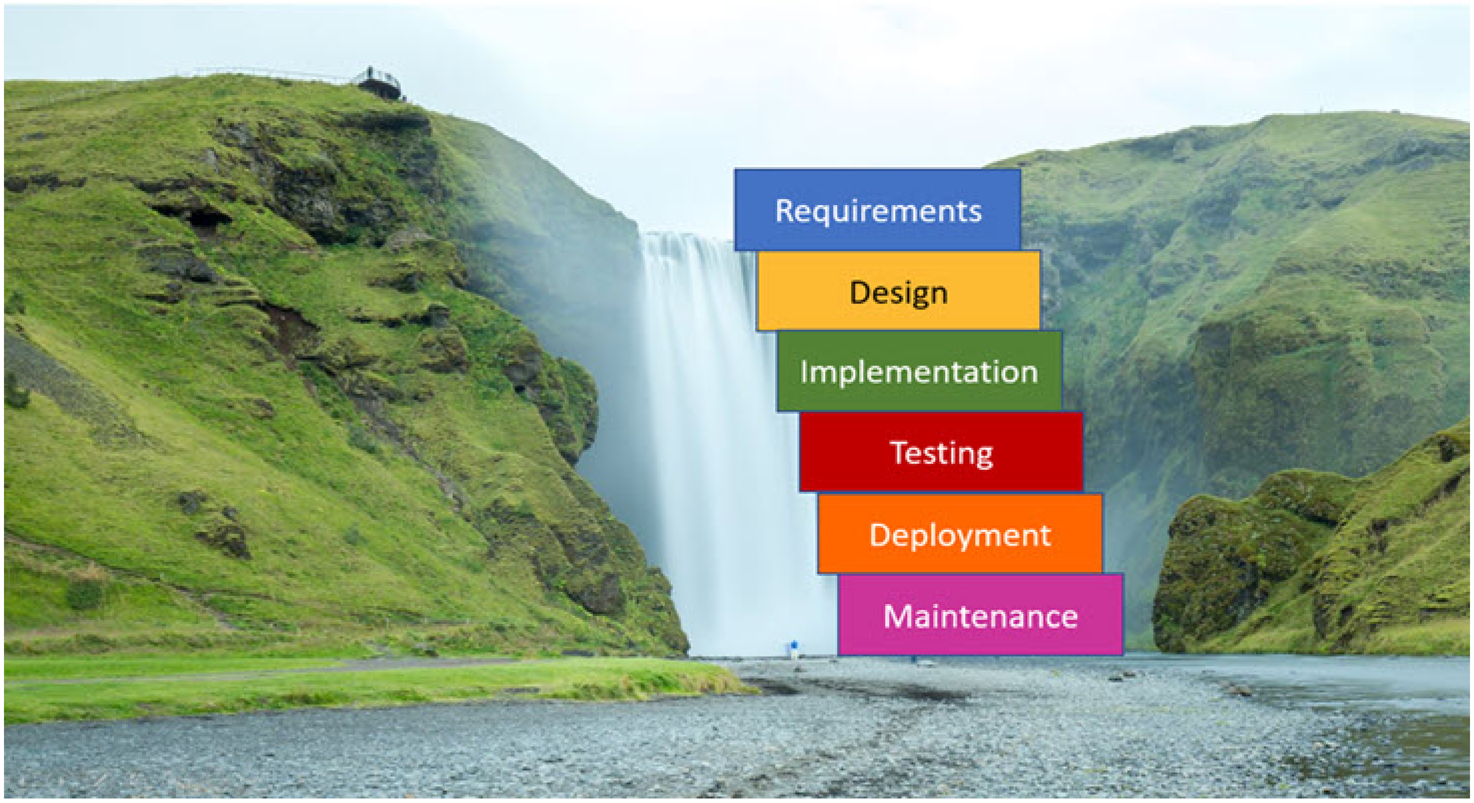
The Waterfall development life cycle model steps are:

Step	Description
Requirements	All requirements for the application being developed are gathered from the client, user, or stakeholder.
Design	The software is documented, diagramed, and designed.
Implementation	The code is written.
Testing	A quality assurance team makes sure requirements are met, the code works properly across devices, and security issues are noted.
Deployment	The application is released to a client or to the public.
Maintenance	The application is monitored for bugs or problems that are patched or fixed while in use. This is an ongoing stage that continues throughout the life of the app.

Understand that an application will likely go through some of these steps multiple times before moving to the next step. For example, the application might go through the Design step five times before it’s ready to move to the Implementation step. Or the application may move back from the Implementation stage to the Design stage if a new feature needs to be added.

This entire development life cycle is a slow process and may take months or years to complete. The Waterfall method also lacks flexibility since the requirements determined in the beginning are carried through to the end product.

Waterfall Model

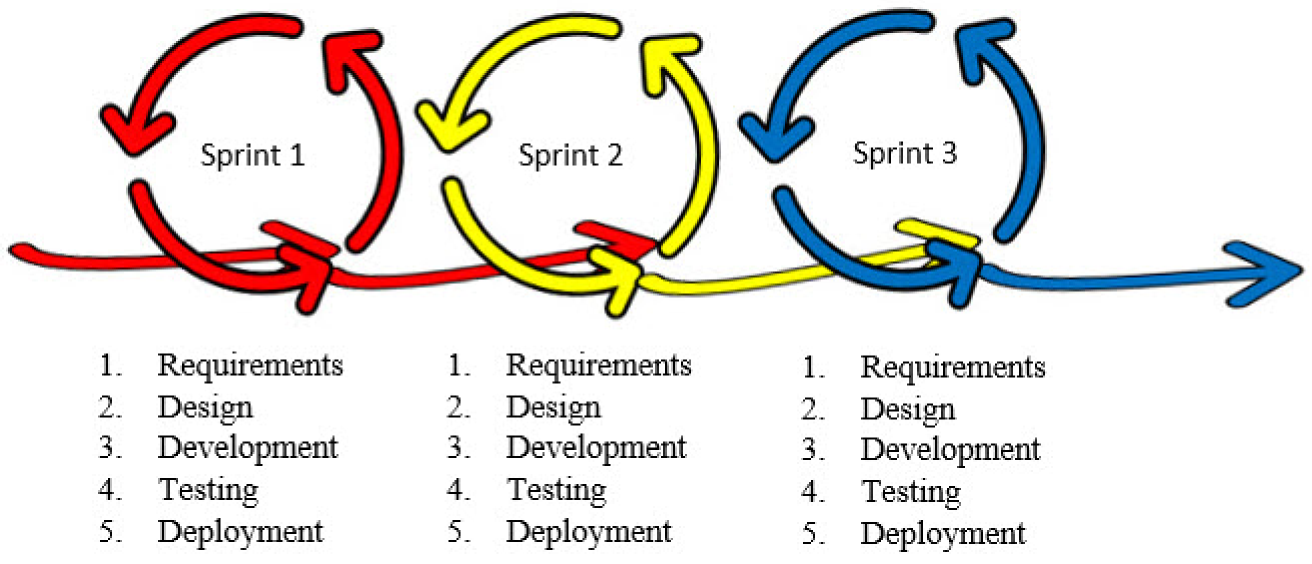


Agile Development Life Cycle Model

A more agile approach was introduced in 2001 that approaches software development as a continuous, changing process with never-ending versions, bug fixes, and enhancements. This approach is aptly named Agile.

The Agile model works in this manner:

- Breaks development into smaller time frames called Sprints
  - Each Sprint has a specific duration (usually two to three weeks)
  - Developers work on one feature during a Sprint
- At the end of each Sprint, developers move on to the next feature
- Testing is performed throughout the development cycle



Coding Errors

Coding errors and design flaws are the main causes of software vulnerabilities. We can categorize these errors into two types:

Coding Error Type	Description
Compile	▪ An error that occurs during the building or compilation stage ▪ Error compromises the software implementation ▪ Prevents app from running
Runtime	▪ An error that occurs while software is running ▪ Sometimes called bugs

Secure Testing Methods

During the coding and design phases of development, you can increase development and application security by implementing a few key practices. Integrate security testing into each step in the development process.

Secure Testing Method	Description
Static application security testing	▪ Known as white box testing ▪ Focuses on analyzing source code, binaries, and byte code early in development process ▪ Good at identifying things like SQL injections and buffer overflows ▪ Can identify exact cause of a coding problem ▪ Only in code that's written but not deployed ▪ Language specific ▪ Can run continually and be widely applied ▪ Has a high percentage of false positives ▪ Limited in the types of vulnerabilities it can detect

Dynamic application security testing	- Known as black box testing - Scans applications after deployment - Tests from the outside - Uses a series of test to determine vulnerabilities and flaws - Not language specific - Has fewer false positives - Hard to automate - Cannot pinpoint the cause of a flaw - Can take up to a week to complete the testing process
Interactive application security testing	Has two types: - Passive - Interactive functionality is built into static application security testing - Uses source code scanners during runtime - Active - Testing tools can access interpreters and compilers, allowing precise identification of a problematic line of code in runtime - Speeds up testing and remediation - Can help in the Development stage by catching vulnerabilities early - Can help in the QA stage by adding automated security checkpoints - Can help in the Production stage through continuous monitoring