Software Engineering

By
Sunil Kumar(Master of Sc.)
Bangalore, India

1. Agenda

- What is software engineering?
- Why is it important?
- What are the building blocks of software engineering
- What are the processes and methods that differentiate it as a discipline?

What is Software Engineering

- Software engineering is the application of a systematic, disciplined, and a quantifiable approach to the development, operation, and maintenance of software, that is, the application of engineering to software.
- One observation to make out of this definition is that software engineering is a discipline applied throughout the entire software lifecycle, spanning the early phases of system specifications, all the way to maintenance after system deployment.

What is Software Engineering

- Why to use an engineering approach to software?
- This is because the alternative would be to use an ad hoc or disordered approach.
- An engineering approach means predictability and quantifiable results through the application of theories, methodologies, frameworks, and tools.
- When applied efficiently, the result is a high-quality software created in a cost-effective manner.

What is Software Engineering

- Software engineering in practice is built of three layers.
  - Process : The process defines the framework and the order around the various activities undertaken in a software project. It sets out how activities or phases, such as requirements, design, construction, and testing are undertaken.
  - Methods : Software engineering methods are practices with proven techniques to perform certain activities. For example, there are methods for analysis and requirement modelling. Similarly, there are methods for design and design modelling, as well as methods for testing.

What is Software Engineering

- Tools : Software engineering tools allow automation of activities, which helps in the systematic application of software engineering. Now in the next sections, let's have a more detailed look at each of these layers.

Process and Process Model

- The process of software creation contains a set of generic activities that can be summarized as follows.
  - Specification and modelling of requirements,
  - Design, and development of the software,
  - Validation of the software,
  - Deployment
  - Software evolution.

Process and Process Model

Process and Process Model

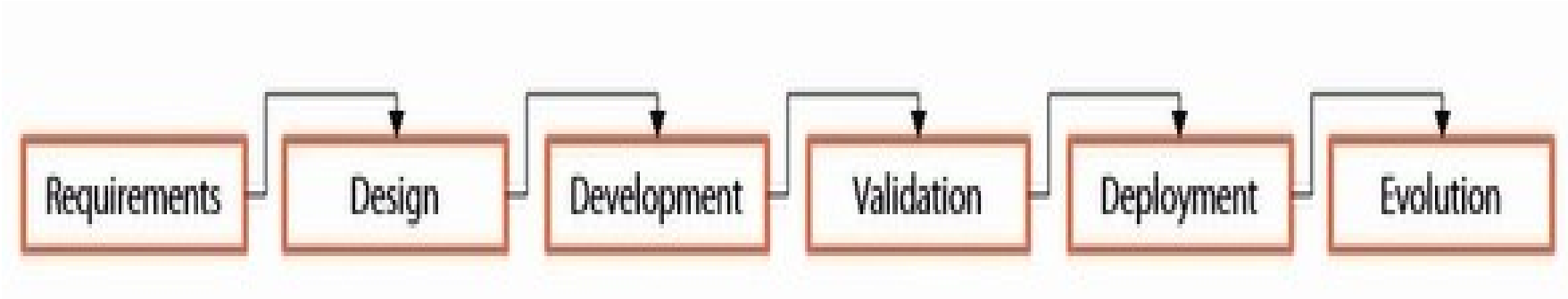
- In addition to these process activities, there are activities commonly referred to as umbrella activities that span the entire software lifecycle.
  - Some of the most common umbrella activities are
  - Project management
  - Quality management or quality assurance,
  - Configuration management,
  - Process improvement.

What is a software process

- A software process, also called a Software Development Lifecycle(SDLC),
- Defines:
  - Tasks inside the Software Engineering activities
  - Order and detail of which these tasks and activities
  - Flow of activities(Iterative, Linear, etc)
  - Type and detail of artifacts

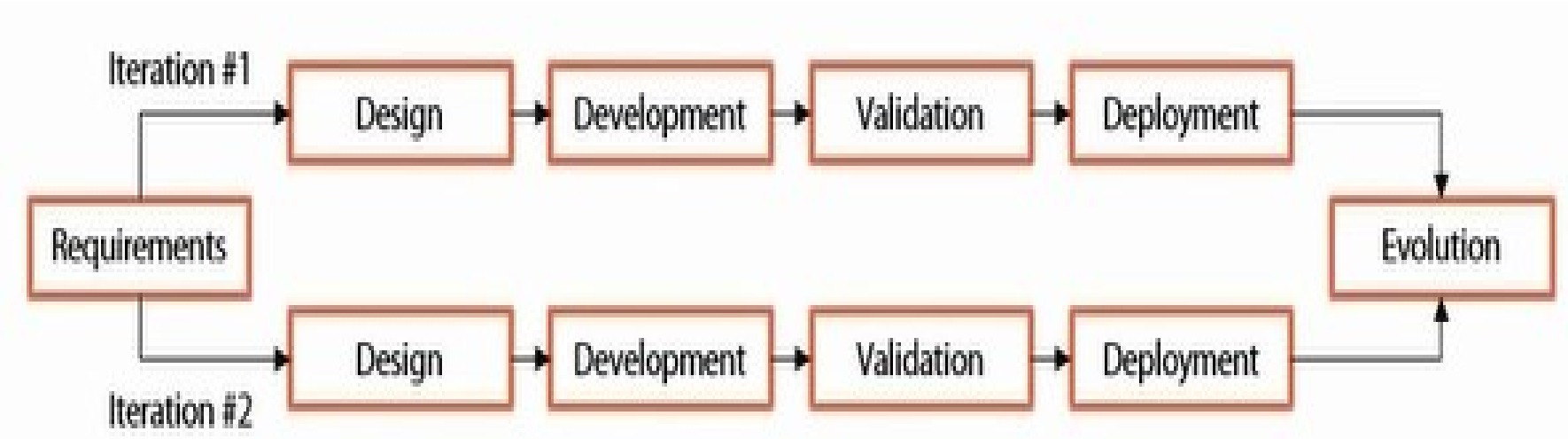
Software Process

- Linear Model
  - Each activity must end before the next one starts



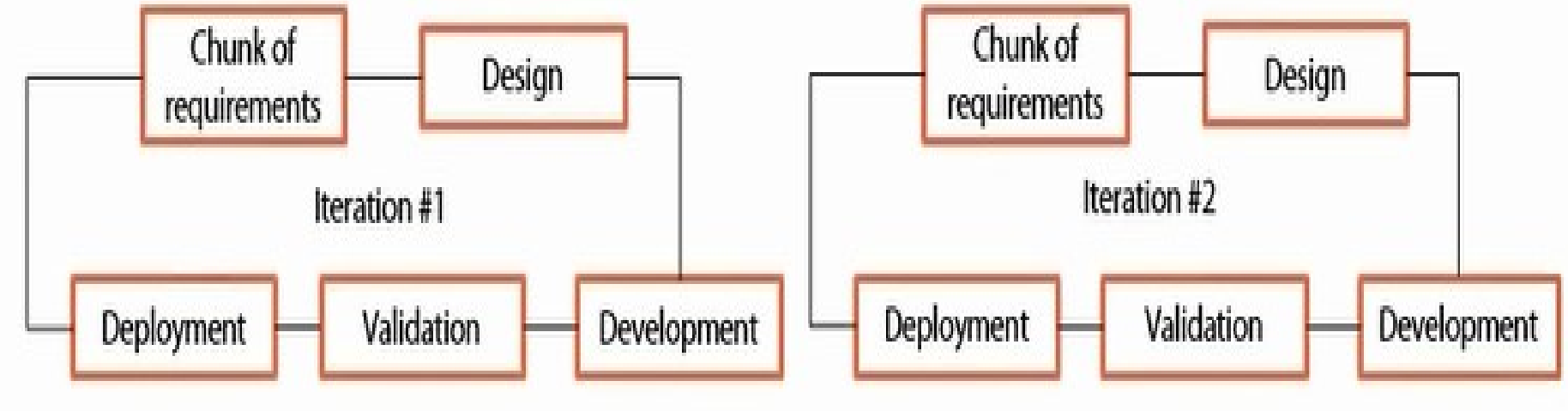
Software Process

- Iterative Model
  - we divide development into multiple iterations, each implementing part of the requirements?



Software Process

- We take this to the extreme by adopting a more agile approach where each iteration has its own requirements to analyze.



Artifacts

- And what about the artifacts? What type of artifacts are we generating, and to what extent of detail? For example, are we generating a lot of requirement and design models, or are we focusing less on design and more on the speed of development. All these decisions are taken based on the software process elected.

Process Model

- So the next question is what are the available models of a software process? Some of the most well-known models are
- Traditional or classical models
  - waterfall, also called linear model
- Iterative and incremental models
  - Prototyping model
  - Spiral model
  - Agile model. I
  - Unified model (combines best practices from both the traditional and the iterative incremental models)

Process Model

- Specialized models (For particular approaches)
  - Component-based development
  - Formal methods
  - Aspect-oriented software development
- Which model is best to use depends on various factors, including an organization's maturity.

- But in general, some projects are more suited to some models. For example, an airplane navigation system requires a very strict and total specification at design, and requires a very detailed level of documentation.

- While on the other hand, a collaboration portal does not need such strict approach, it requires continuous customer feedback, and requires far less detailed documentation.

- Typically the project manager, in coordination with a lead software engineer, decides on the process model most appropriate for a specific project. There is one important note that I want to highlight. It's often the case that software professionals refer only to software creation when they talk about the software process, and they neglect the evolution or maintenance part, meaning that when they design the software process, they think about requirements through deployment, but forget about the maintenance phase.

- This is wrong because the maintenance, or evolution phase, is part to the overall lifecycle of a system until the system is retired. Therefore when you plan the software process to use, make sure to plan the overall lifecycle. At the end keep in mind that changes that are requested to a system after deployment are also requirements that need analysis, design, development, testing, and deployment. The only difference is that these requirements are often requested as part of a change management process and are not driven by project charter.