Software Engineering

Lecture 1.1
Introduction

SAURABH SRIVASTAVA
ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
IIT (ISM) DHANBAD

What is Software?

"Software is: (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to adequately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs."

Software Engineering – A Practitioner's Approach, 8th Edition

[**Roger S. Pressman** and **Bruce R. Maxim**]

Programs

The first element of the definition here is the set of “executable instructions”

The more common term that we use for these instructions is a *Program*

- Another colloquial term for the same is *Code*

Traditionally, the people in an organisation that “wrote code” are called “coders” or “developers”

- Some other roles that could be present in the organisation included “testers” and “system administrators”
- However, these distinctions are fading away quickly (check your Homework)

Although, these terms may imply that programs are written only by developers, it is not true

- Testers as well as System Administrators may also “write code” …
- … it is just that the purpose of this code differs from the purpose of the code written by developers

To summarise, in a typical software environment, we may have “different kinds” of programs

- While some of them achieve what can be seen as the external output of the environment …
- … others may make sure they work “as expected”

Data Structures

The second element of the definition addresses needs pertaining to processing *Data*

- Data refers to the information that the product ingests, manipulates and outputs

To store and process data, programs rely on some “information skeletons”

- These skeletons are often referred to by an umbrella term called *Data Structures*

Data structures capture the properties of the real-world data

- For example, a Class may be represented by a set of Students …
- … which in turn, can be represented by a collection of integers, representing their Roll Numbers

Data structures, thus, map real-world information to elements that a Computer can “understand”

- For instance, a Computer may not understand what a ”Roll Number” means …
- … but it does understand how to store and process an integer

Thus, data structures provide a layer of mapping between elementary data types and complex data

Documentation

The third element of the definition talks about *Documents*

- To be precise, documents about the Software

The most common objective to write these documents is to explain the working of the code

- For example, a document called a *Call Graph* may explain how one piece of code interacts with another

Other documents may attempt to capture the overall structure of the code

- For example, the *Logical View* of a software system depicts its major sub-systems …
- … and probably sub-sub-systems too !!

There could be many other documents too

- This may include *User Manuals*, *Test Plans*, *Deployment Guides* etc.

Overall, documents enable us to reason about, and effectively use, the built software product

What is Software Engineering?

"Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1). "

IEEE Standard Glossary of Software Engineering Terminology

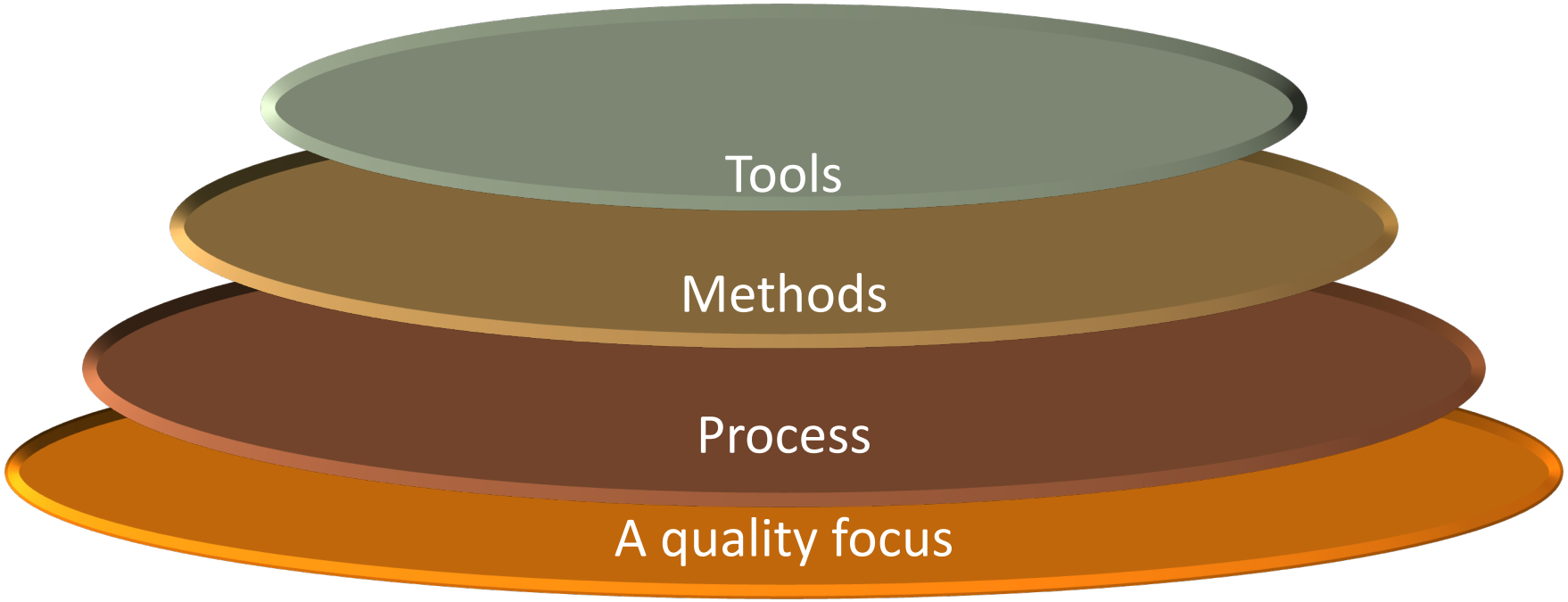
[**IEEE Standard 610.12-1990**, Page **67**]

“… application of engineering to software …”

Although the exact definition of Software Engineering is not of much interest …

- … the part “… the application of engineering to software …” is an important aspect
- It means that building software is a systematic procedure that requires following a disciplined approach

Software Engineering may be envisioned as a Layered Activity



Software Engineering Layers

Software Engineering – A Practitioner's Approach, 8th Edition

[**Roger S. Pressman** and **Bruce R. Maxim**]

“… application of engineering to software …”

Although the exact definition of Software Engineering is not of much interest …

- … the part “… the application of engineering to software …” is an important aspect
- It means that building software is a systematic procedure that requires following a disciplined approach

Software Engineering may be envisioned as a Layered Activity

- The base for all engineering tasks is a pledge to build *quality* products – a software product is no different
- A Software Engineering *Process* is the systematic approach that is followed to build the product …
- … it also includes (agreed upon) documents to produce, alongside the actual product
- A particular Software Engineering Process may ask the practitioners to adopt some *Methods* …
- … as part of the development, e.g., Enlisting Requirements, Documenting Architectures, Testing etc.
- There are existing software products, which can be used as *Tools* to build other software products …
- … for example, Workflow Designers, Documentation Helpers, IDEs, Testing Tools etc.

Developing Software

There are many steps involved in building commercial software products

While the actual number of steps discussed in literature vary, there are three major phases

- These phases usually happen in a cycle over multiple iterations

In the first phase, we analyse the problem at hand, and prepare prospective plans

- Typically, we analyse if the project is feasible subject to the available people and resources
- If so, we chalk out detailed plans to for the complete development process
- We also prepare some sketches and documents, which can guide the upcoming phases

Next, we begin the process of implementation

- This may involve finding and using suitable libraries or reusing code fragments from previous projects
- It usually also involves writing new code fragments to supplement existing parts

Last, we release or deploy the software, and provide support to the users

- The process is also called as “Maintenance” – aka keeping the software free of problems

Software Development Lifecycle – 1/2

The Software Development Lifecycle (SDLC) is a common term used to refer to the development process

- The actual steps of development may not walk through these phases religiously…
- … but the general development process may resemble SDLC closely

The Planning Phase

- We begin by analysing the requirements of the project and evaluating if they are feasible
- Two most important aspects of feasibility are the development cost and skills of the developers
- The next step is to create some initial design – a bird's eye view of the software
- The designing process continues till we have "sufficiently detailed" design documents

Software Development Lifecycle – 2/2

The Implementation Phase

- Implementation refers to actually developing the software at the level of code and binaries
- This includes writing new code, reusing code from previous projects and using third-party libraries…
- … to achieve the functionality expected from the software
- The development is tightly coupled with Testing – the process of checking code fragments for problems
- Testing is performed at lower (or Module) levels as well as higher level (for the system or a group of Modules)

The Maintenance Phase

- At the end of development, the software is finally deployed or released
- But the job of the development team doesn't end there – it is only the beginning of a long journey
- Throughout a software's planned lifecycle, the developers keep performing Maintenance
- Maintenance may be performed to weed out existing problems…
- … or, to enhance the software with new features

The Waterfall Model of development

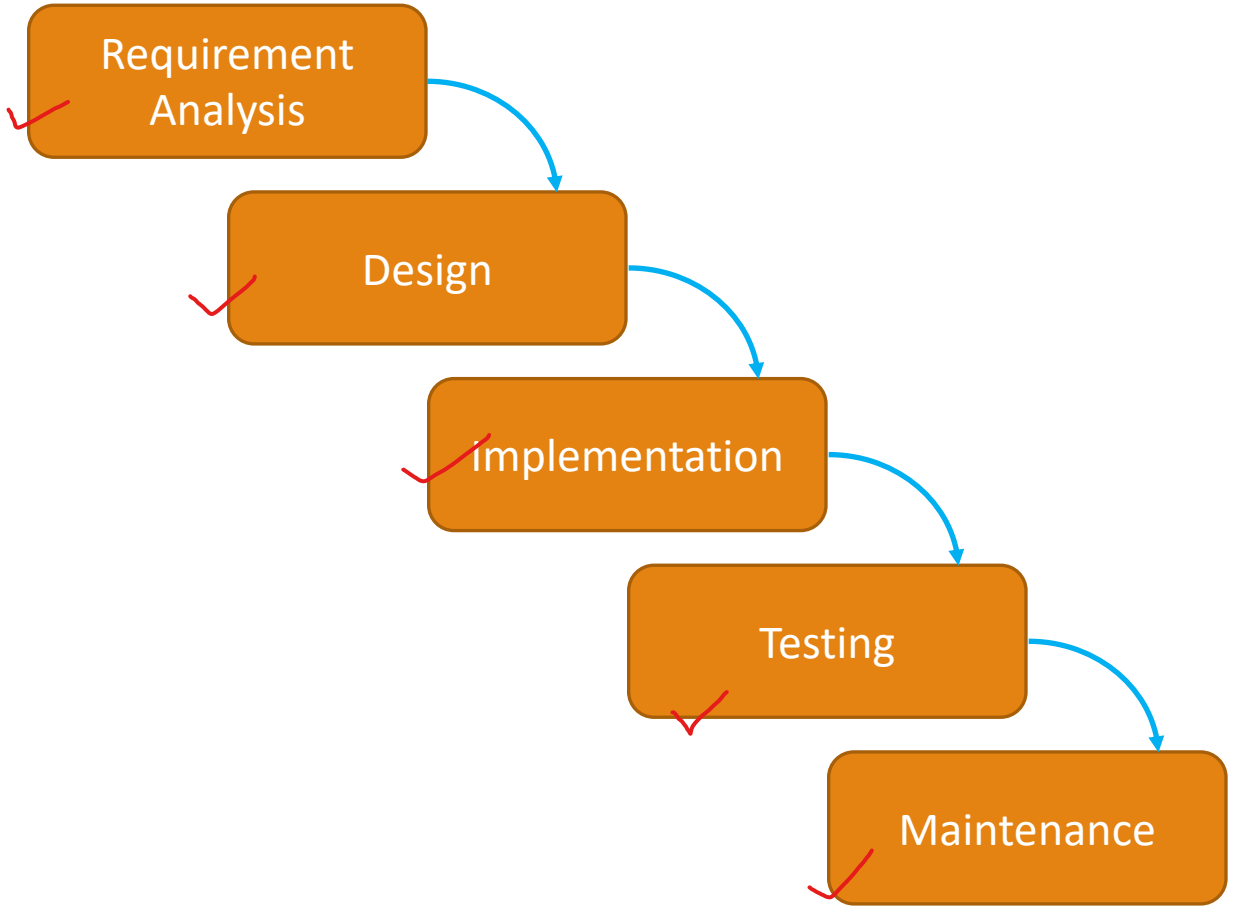
The earliest method of developing software was probably the Waterfall Model

- In terms of the Layered vision, the Waterfall model is actually a Software Engineering Process

The Waterfall model defines the steps involved in software development in a sequential order

- It is akin to the fall of water from the top to bottom, hence, the “Waterfall” model

The Waterfall Model of development



The Waterfall Model of development

The earliest method of developing software was probably the Waterfall Model

- In terms of the Layered vision, the Waterfall model is actually a Software Engineering Process ✓

The Waterfall model defines the steps involved in software development in a sequential order

- It is akin to the fall of water from the top to bottom, hence, the “Waterfall” model

The steps of the Waterfall model generally include

- Requirement Analysis – collecting and analysing the requirements related to the software
- Design – coming up with an Architecture for the software (a blueprint of the finished product)
- Implementation – writing or absorbing code fragments from third-parties to achieve functionality
- Testing – testing the software at module level, after inter-module integrations and the overall software
- Maintenance – providing post-deployment support for the software

Homework

Pick up a Software Engineering book

- The book colloquially called “Pressman” is the one I will follow from time to time (*Software Engineering – A Practitioner’s Approach*, 8th Edition, **Roger S. Pressman** and **Bruce R. Maxim**)
- When you have time (e.g., when you are not able to sleep in the night :-P) …
- … skim through the first three chapters of the book (there’s no hurry, you can do it a day before exam too :-D)