Software Engineering

Lecture 1.2

Software Development Overview

SAURABH SRIVASTAVA

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

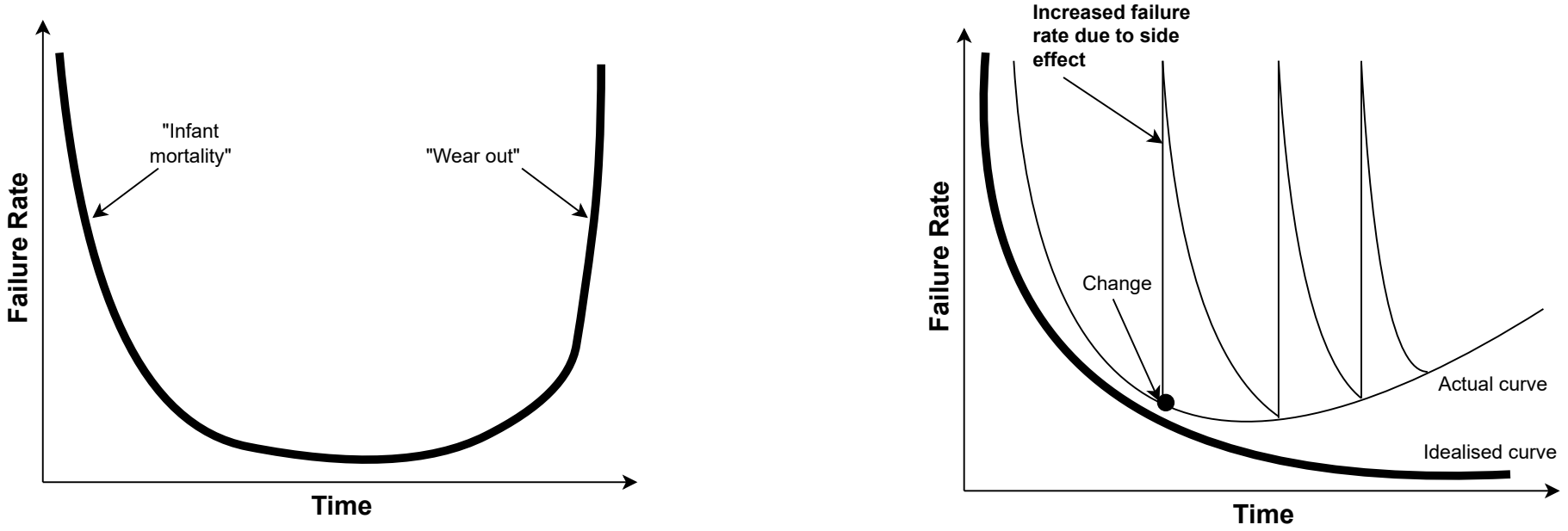
IIT (ISM) DHANBAD

Before we move ahead …

In the first Lecture I told you about a failure curve

- It is time to have a look at it before we go any further

The Hardware Analogy for Software



Failure Curves for Hardware (Left) and Software (Right)

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

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

Before we move ahead …

In the first Lecture I told you about a failure curve

- It is time to have a look at it before we go any further
- The curve shows that the “expectations” from software to have an idealized failure curve …
- … e.g., one where failures do not occur due to “wear out”, are rather wishful

The biggest challenge with a software is to keep up with “changes”

- Usually, for hardware, changes are infrequent (especially when it comes to changes in fundamental behaviour)
- But the expectations from the user that a software product can be changed “as often as required” …
- … makes it prone to degradation over time, and an eventual stage where it has to be rebuilt from scratch

Legacy software is a term that is used to refer to software applications which were built “far back”

- Here, ”far back” usually refers to many decades
- Such software are often maintained by businesses to avoid a large amount of rework …
- … but they may require significant changes in their architecture to adopt to modern requirements

Software Application Types

Stand-alone/Window Applications

- Built for a specific platform (e.g., Windows or Mac) for a specific architecture (e.g., x86 or x86_64)
- Significant effort may be required to port it to another platform

Web-based Applications

- Execute over browsers; accessible over a wide range of platforms
- Some effort may be required to cater to different “screen sizes” (e.g., for different mobile devices)

Mobile Applications

- Similar to Window Applications, but for Mobile Platforms (e.g., Android/iOS)
- Could also be HTML5 apps (similar to Web-based Applications)
- Major effort may be required to port it from one platform to another (e.g., from Android to iOS)

Off-the-shelf vs Bespoke Software

Some software products are available off-the-shelf

- You can purchase/download them and put to use straightaway
- These products often have no/minimal customisability
- The best example includes Operating Systems, Office Suites, Numerical Computation Tools etc.

Some organisations may want solutions tailored to their needs

- It may be because no off-the-shelf solution fulfils their requirements fully
- In these cases, they raise bids for custom software development, also called *bespoke* development
- The best example includes software stacks of different Banks and custom websites for organisations

For most part of this course, the software development that we will talk about will be *bespoke*

- This is because it involves a large number of activities, and you can appreciate the complete process

The idea of a Software Process

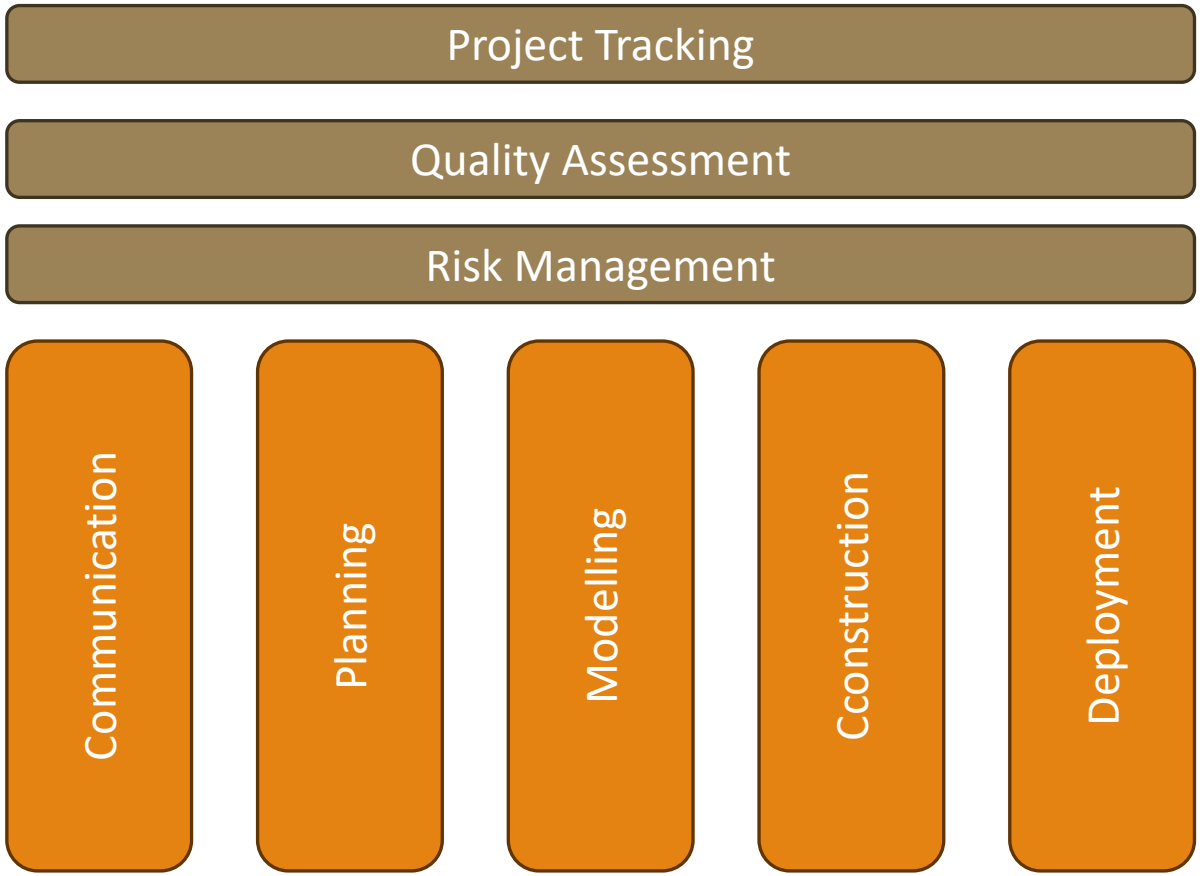
A Software Process can be seen as a collection of *activities*

- An activity is a generic term for a phase of software development
- While certain activities occur during a particular phase of the development, …
- … others continue throughout the complete development cycle
- Activities which continue throughout the development cycle are called *umbrella activities*
- For our purpose we will call the other activities as *core activities*

Any Software Development process essentially contains the core and umbrella activities

- Albeit some of the activities may occur in an iteration
- Pressman et. al. consider five core activities – *communication*, *planning*, *modelling*, *construction*, *deployment*
- Umbrella activities include *risk management*, *quality assurance* and *project tracking*

Generic Activity Framework



The idea of a Software Process

A Software Process can be seen as a collection of *activities*

- An activity is a generic term for a phase of software development
- While certain activities occur during a particular phase of the development, …
- … others continue throughout the complete development cycle
- Activities which continue throughout the development cycle are called *umbrella activities*
- For our purpose we will call the other activities as *core activities*

Any Software Development process essentially contains the core and umbrella activities

- Albeit some of the activities may occur in an iteration
- Pressman et. al. consider five core activities – *communication*, *planning*, *modelling*, *construction*, *deployment*
- Umbrella activities include *risk management*, *quality assurance* and *project tracking*

This framework can be seen as a generic template

- Different Software Development Processes can be considered as instances of this template
- **The next unit is dedicated to studying these instances and their differences with each other**

The Agile Philosophy

In 2001, a document titled the “Manifesto for Agile Software Development” was released

- It can be considered as a watershed moment in the history of Software Development

The cover page of the document itself summarises the philosophy

- “… *Individuals and interactions* over *processes and tools* …”
- “… *Working software* over *comprehensive documentation* …”
- “… *Customer collaboration* over *contract negotiation* …”
- “… *Responding to change* over *following a plan* …”

The philosophy has both merits and problems

- The biggest merit of the philosophy is that it represents a more realistic view of software projects …
- … i.e., scenarios where plans and processes have to be compromised over working deliverables
- The major problem with it is that it is often cited as an excuse for “laziness” and “lack of professionalism”
- This leads to products that are a nightmare to maintain with little or no documentation

SDLC Documentation Overview (1/2)

Documentation is an important – yet often ignored – task in SDLC

- People who practice Agile methodologies are often guilty of ignoring documentation

For bespoke development projects, the first produced document highlights major requirements

- This document, called the *Request For Proposal* or the RFP in short
- It provides detailed "expectations" of the user from the solution they seek
- This in turn invites proposals from multiple software development vendors
- RFP may contain requirements that "may not be achievable" by any vendors
- During a phase of interaction with vendors, the user may revise the expectations to make them doable

After the due process, one vendor is chosen as the organisation to carry out the development

- The next document in the pipeline is the *Business Requirements Document* or the BRD
- BRD is a more refined form of RFP, containing only those requirements that the vendor agreed to
- It may also contain legal terms and conditions to which both parties must adhere
- It may also include information for staffing, schedules and payment

SDLC Documentation Overview (2/2)

In the next phase, a series of interactions occur between the stakeholders

- These meetings are centred around understanding finer details of all the Requirements
- This includes both functional and non-functional requirements of the solution
- The process culminates into creation of a document called the *Software Requirements Specifications* or SRS

In some cases, another document may be produced to capture even finer details of the Requirements

- This document is called the *Functional Requirements Specification* or FRS
- FRS contains implementation-level information for implementing expected features from the solution
- It is useful to both the developers as well as the testers of the system

Some other documents that may be produced in the SDLC includes

- Architecture and Design documents – detailing the structure of, and interactions happening in the system
- User Manuals – documents to guide users about the usage of the system
- Documents related to Quality Assurance – we will now discuss them in a little more detail !!

In the next Lecture …

I hope you now have an idea of what is coming up

- In short, we will talk about different software development activities over various units

The first target is to get to know some examples of Software Development Processes

- We start with in the next Lecture